THE UNIVERSITY
of EDINBURGH

Practical Differential Privacy in High Dimensions

Daniela Antonova

Master of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2015

Abstract

Privacy-preserving, and more concretely differentially private machine learning, is concerned with hiding specific details in training datasets which contain sensitive information. Many proposed differentially private machine learning algorithms have promising theoretical properties, such as convergence to non-private performance in the limit of infinite data, computational efficiency, and polynomial sample complexity. Unfortunately, these properties have not always translated to real-world applications of private machine learning methods, which is why their adoption by practitioners has been slow. For many typical problems and sample sizes classification accuracy has been unsatisfactory. Through feature selection which preserves end-to-end privacy, this work has demonstrated that private machine learning algorithms can indeed be useful in practice. In particular, we propose a new feature selection mechanism, which fits well with the design constraints imposed by differential privacy, and allows for improved scalability of private classifiers in realistic settings. We investigate differentially private Naive Bayes and Logistic Regression and show non-trivial performance on a number of datasets. Significant empirical evidence suggests that the number of features and number of hyperparameters can be determining factors of the performance of differentially private classifiers.

Acknowledgements

I am deeply grateful to my supervisors, Dr. Iain Murray and Dr. Miles Osborne, for giving me a chance to make a scientific contribution and for the extraordinary learning experience that has been the past two years. I would like to especially thank Dr. Iain Murray for his continuous support, inspiration, and excellent example. I am forever indebted for the opportunity to work with the kind of researcher I can only dream of becoming.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Daniela Antonova*)

Table of Contents

Chapter 1

Introduction

Few issues in this new era of Big Data have been dogged by as much controversy as privacy. Arguably, this basic human right is being destroyed by governments and private institutions alike. “Relying on the government to protect your privacy is like asking a peeping tom to install your window blinds,” says John Perry Barlow, voicing a growing public concern.

With great data comes great responsibility, but many institutions have failed to carry it out. While data has been successfully utilized in numerous domains to improve quality of life, publishing results of data analytics has not been completely uneventful.

Many attempts to solve real world problems with machine learning have resulted in privacy breaches, the most cited of which is the de-anonymization of the Netflix user movie ratings dataset (Narayanan and Shmatikov, 2008). The attempt by Netflix to improve its recommendation engine resulted in a million-dollar lawsuit (Newman, 2009) and all following competitions getting cancelled. Other incidents include the AOL search data leak (Wikipedia, 2004), a genome wide association study identity and disease leak (Wang et al., 2009), the Taxicab passenger privacy leak (Atockar, 2014), and personal data leaks in online advertising (Korolova, 2010).

Privacy leaks like these result from the intrinsic peculiarity of high-dimensional data: such samples are unique too often. With just a little side information it was possible to uniquely identify individuals in the above datasets, even though obvious identifiers, such as names, social security numbers, address, etc. had been removed. Consider the linkage attack on the Netflix dataset. The attackers correlated the movie ratings in a publicly available IMDb dataset with the movie ratings in the Netflix dataset and matched with high accuracy the anonymized records from the Netflix dataset with the public profiles in the IMDb dataset.

To combat privacy attacks which rely on background information, some more robust formal definitions of privacy have been formulated. Work on privacy of statistical databases started with the definition of statistical disclosure, and progressed to data scrubbing, and k-anonymity (Sweeney, 2002) and its variants, including l-diversity (Machanavajjhala et al., 2007) and t-closeness (Li et al., 2007). All of these approaches group samples by their sensitive attribute. Privacy is then satisfied only if the number of samples in each group is large enough. Another common denominator is the approaches' modelling of the possible auxiliary information an attacker could possess. When the assumptions break, so does the privacy guarantee of these methods, failing to prevent the above privacy leaks.

Differential privacy is an emerging definition which dictates that the presence or absence of an individual from a database cannot affect the released statistics significantly. This formal notion of privacy avoids side information assumptions by requiring privacy in the worst case scenario: an attacker knowing all but one entry in a database. In this way, independent of the auxiliary information available to an attacker, they are unable to discern the participation of an individual in the database.

A private algorithm, also referred to as a mechanism, guarantees differential privacy by returning a noisy version of a non-private algorithm's output. For example, if the private mechanism counts fingers on a hand it might return four instead of five. Privacy is achieved by adding a noise vector from a certain distribution, such that the private output is relatively close to the true one, in some sense. If only a little noise is added, the privacy restrictions are relaxed, and the output is very close to the true one. A user of the private algorithm's output can draw similar conclusions from the data, as they would from the non-private algorithm. On the other hand, if the noise is large in magnitude, the privacy guarantee is stricter, but the utility of the data deteriorates. This privacy-utility trade-off is the cornerstone of practical private mechanisms.

High-dimensional data offers many challenges for machine learning, and even more so for privacy-preserving machine learning. Existing differentially private mechanisms struggle with high-dimensional data (Stoddard et al., 2014; Dwork et al., 2014; Ji et al., 2014; Fredrikson et al., 2014). When the input dataset to a randomized algorithm (a learning procedure, for example) has a large number of features, a prohibitive amount of noise compared to the signal in the data has to be added, rendering it close to useless.

Diminished accuracy as a result of privacy restrictions can be naturally overcome with additional data. Unfortunately, depending on the dimensionality of the dataset and the stringency of the privacy requirement, the amount of extra data required for good

utility ranges from significant to exorbitant (see Appendix B).

This work aims to evaluate and extend applications of differential privacy to machine learning. The focus is on binary classification problems where the dimensionality of the data is non-trivial and the amount of data is insufficient for high-accuracy private results using existing algorithms.

Much of the prior work on privacy preserving computation has focused on the release of privatized data. After perturbing the data itself, standard machine learning technology could then be employed making use of it, producing a private model. However, for most real world datasets, the amount of noise added to the data is intolerable (Brickell and Shmatikov, 2008). Instead, later approaches, and the approach taken in this work, integrate the privacy guarantees into the algorithm, ensuring that the algorithm is private while still using raw data. This strategy is underpinned by the principle that the smallest amount of information has to be released if the private mechanism is to be effective. Instead of publishing the whole dataset, only the model parameters are published, considerably less information.

1.1 Our contributions

Most research in differential privacy in machine learning has created counterparts of existing machine learning methods, but the scalability properties of these methods to real high-dimensional datasets remain to be investigated. This work begins to provide insight into these practical issues by focusing on basic differentially private classification approaches. We demonstrate the successful use of the differentially private Naive Bayes classifier with stringent privacy guarantees on several high-dimensional datasets. We further compare these results with the negative results of private logistic regression. In particular,

- We provide a review section which aims to make differential privacy more approachable to practitioners, where we give several intuitive and rigorous justifications for composition theorems, as well as the generality of the Exponential Mechanism, a standard approach to creating differentially private algorithms.

- We develop, theoretically justify, and empirically evaluate differentially private Naive Bayes Maximum Likelihood Estimation. We employ the established Laplace mechanism and show that private Naive Bayes scales poorly to high-dimensional real-world datasets.

- To address this issue, we employ univariate differentially private feature selection. We detail several naive approaches to the problem and justify their low scalability with the dimensionality of the problem. We further develop a new univariate feature selection technique based on dynamic programming and the Exponential Mechanism for which the privacy guarantees degrade with the number of features to be selected, rather than the original dimensionality of the problem.

- We further develop an original differentially private feature selection heuristic based on a low sensitivity count-based metric, which, together with the feature selection algorithm, achieves significantly better results than private Naive Bayes on its own. We also provide comparisons with the private version of the standard univariate feature selection technique of using the mutual information of each feature with the class label and show that the count-based heuristic has significant advantages.

- Finally, we analyse and empirically evaluate several approaches to private logistic regression, showing their poor performance on real-world high-dimensional datasets.

Chapter 2

Differential Privacy and Machine Learning

The search for a formal framework in which to develop privacy-preserving algorithms has circled in on differential privacy. This is a condition on a randomized algorithm which requires that the participation of any individual in an input dataset should not alter the probability of any outcome of the algorithm too much. Since the output does not change significantly it cannot be used to infer any individual's data, or even the participation of the individual if the data is known. We now state the definition of differential privacy more formally, after setting up some necessary notation.

2.1 Definition

In the privacy literature, mechanism is a generic term for a private randomized algorithm. It could be any query to a database, a training procedure, or any other algorithm whose randomized output protects the privacy the original data.

Notation The notation used here follows Hall (2013) and is summarized in table 2.1. Let $\mathcal{X}$ be the space of all possible datasets. A (non-private) algorithm on datasets in $\mathcal{X}$ can be characterized in terms of the function it outputs. A deterministic algorithm produces a vector $\boldsymbol{\delta}_X = \delta(X)$ of a finite dimensionality D on a dataset X, whose elements could be real numbers, integers, or binary. The algorithm is completely described by the set of vectors $\{\boldsymbol{\delta}_X : X \in \mathcal{X}\}$. For each possible dataset, a randomized algorithm induces a distribution over the output vectors in $\mathbb{R}^D$. This distribution is referred to as P_X, when the input dataset is X. Thus, a randomized algorithm is characterized by the distributions $\{P_X : X \in \mathcal{X}\}$.

Notation	Meaning
$\mathcal{X}$	the set of all possible datasets
X, X'	two datasets in $\mathcal{X}$; this is the design matrix augmented with the label vector
$X \sim X'$	two adjacent (aka neighbouring) datasets in $\mathcal{X}$: X' was created by taking all samples of X and adding one new sample
$\lvert X \ominus X' \rvert$	the set of all *examples* that are either in X or in X', but not both
$\mathcal{P}$	the private mechanism
$\{\boldsymbol{\delta}_X : X \in \mathcal{X}\}$	all possible outputs of a non-private algorithm $\mathcal{P}$
$\{P_X : X \in \mathcal{X}\}$	all possible distributions induced on the output space by mechanism $\mathcal{P}$
$\mathbf{z}$	a sample from the distribution characterizing the private mechanism
$\mathcal{Z}$	the range of the private mechanism
$P_X(\mathbf{z})$	$P(\mathcal{P}(X) = \mathbf{z})$ or $P_X(\mathbf{z} \in \mathcal{Z})$, the probability of the private mechanism $\mathcal{P}$ producing output $\mathbf{z}$ on a dataset X
$dP_X(\mathbf{z})$	the probability density characterizing the distribution P_X of mechanism $\mathcal{P}$ on dataset X
ε	privacy parameter
Δ	global sensitivity
$d(\mathbf{z}, X)$	the distance metric of the exponential mechanism
N	number of examples in the dataset
D	number of dimensions in the dataset
$\mathcal{L}(\boldsymbol{\theta})$	likelihood function

Table 2.1: Summary of notation

The output of a private algorithm is then a sample from P_X. We will use $P_X(\mathbf{z})$ to mean the probability of the private mechanism $\mathcal{P}$ producing output $\mathbf{z}$ on a dataset X. In other words,

$$P_X(\mathbf{z}) = P(\mathcal{P}(X) = \mathbf{z}). \tag{2.1}$$

Finally, differential privacy is concerned with what happens to the output of a private mechanism when the input changes. To formally define the privacy guarantee, we need a formal definition of the distance between two datasets.

Definition 2.1.1. Two datasets X and X' are said to be adjacent or neighbouring whenever they differ in one example, i.e. X' was created by taking all samples in X and adding one new sample. Thus the set of all samples that are in X' but not in X has cardinality 1, $|X' \ominus X| = 1$, and the set of all samples that are in X but not in X' is empty, $|X \ominus X'| = 0$.

We are now ready to state the definition of differential privacy.

Definition 2.1.2. A randomized algorithm $\mathcal{P} = \{P_X : X \in \mathcal{X}\}$ on the probability space $(\omega, \mathcal{Z}, P)$ satisfies ε-differential privacy if for any two adjacent databases X and X' and for all events $\mathbf{z} \in \mathcal{Z}$

$$e^{-\varepsilon} \leq \frac{P_X(\mathbf{z})}{P_{X'}(\mathbf{z})} \leq e^{\varepsilon}. \tag{2.2}$$

In this definition the distribution is over the choices made by the randomized algorithm. The ratio is interpreted to be 1 if $P_X(\mathbf{z})$ and $P_{X'}(\mathbf{z})$ are both 0.

Interpretation For an algorithm, being differentially private means that it protects its input dataset against an adversary who knows all but one of its samples. In that sense differential privacy is a worst-case guarantee, since there is no upper bound on the amount of information an attacker can possess.

To understand how privacy of a sample is preserved under this guarantee, consider the hypothesis test performed by the attacker, whose goal is to find out the values of the last sample in the database. The hypothesis test was first laid out by Wasserman and Zhou (2009) and is carried out as follows. Two hypothesis H_0: X and H_1: X' are tested, where only X' contains the sample in question. The power of the hypothesis test is controlled by the privacy parameter (Wasserman and Zhou, 2009), (Oh and Viswanath, 2013). Let $\mathcal{Z}$ be the set of outputs of the private mechanism for which H_1 is rejected and $\mathcal{Z}^C$ be the set of outputs for which H_0 is rejected. Then the false alarm probability

and the missing detection probability are

$$P_{FA} = P(\mathcal{P}(X) \in \mathcal{Z}^C) \tag{2.3}$$

$$P_{MD} = P(\mathcal{P}(X') \in \mathcal{Z}) \tag{2.4}$$

Applying the differential privacy definition 2.1.2, we get

$$1 - P_{FA} = P(\mathcal{P}(X) \in \mathcal{Z}) \leq e^{\varepsilon} P(\mathcal{P}(X') \in \mathcal{Z}) = e^{\varepsilon} P_{MD} \tag{2.5}$$

or

$$e^{\varepsilon} P_{MD} + P_{FA} \geq 1. \tag{2.6}$$

Similarly,

$$P_{MD} + e^{\varepsilon} P_{FA} \geq 1 \tag{2.7}$$

can be obtained by switching the two datasets in definition 2.1.2. Since $1 - P_{MD}$ is the true positive rate, it is also possible to draw the ROC curve for the hypothesis test ($1 - P_{MD}$ as a function of the false positive rate).

A relaxed definition of differential privacy also exists, which allows the failure of the exact differential privacy criterion with probability at most δ, where the probability is due to the randomized algorithm.

Definition 2.1.3. A randomized algorithm $\mathcal{P} = \{P_X : X \in \mathcal{X}\}$ on the probability space $(\omega, \mathcal{Z}, P)$ satisfies (ε, δ)-differential privacy if for any two adjacent databases X and X' and for all events $\mathbf{z} \in \mathcal{Z}$

$$P_X(\mathbf{z}) \leq e^{\varepsilon} P_{X'}(\mathbf{z}) + \delta. \tag{2.8}$$

In this work we focus on exact differential privacy, but many of the analysis presented here can easily be extended to the approximate differential privacy guarantee.

To sum up, the differential privacy constraint means that the false alarm probability and the missing detection probability cannot both be too small. Thus, in essence, the differential privacy guarantee quantifies how difficult the attacker's inference would be.

2.2 Basic Mechanisms

This section details established techniques for performing differentially private computations on sensitive data. We discuss the private release of D-dimensional vectors. There are several ways of ensuring that a private mechanism $\mathcal{P}$ satisfies ε-differential privacy and we detail the most frequently used below.

2.2.1 Laplace Mechanism

Consider a non-private algorithm which outputs a vector $\boldsymbol{\delta}$. A private mechanism could report this vector by perturbing it in some direction, by adding a noise vector drawn from a Laplace distribution with an appropriate scale and zero mean. Such perturbation is the basis of the Laplace mechanism, which achieves ε-differential privacy by scaling the noise distribution according to the privacy parameter. The resulting vector $\mathbf{z}$ preserves ε-differential privacy.

Theorem 2.2.1. The Laplace mechanism is ε-differentially private:

$$dP_X(\mathbf{z}) \propto \exp\left(-\frac{\varepsilon}{\Delta}\|\mathbf{z}-\boldsymbol{\delta}_X\|_1\right), \tag{2.9}$$

where

$$\Delta = \max_{X\sim X'} \|\boldsymbol{\delta}_X - \boldsymbol{\delta}_{X'}\|_1 \tag{2.10}$$

is the *global sensitivity*, defined as the maximum change in the output of the non-private algorithm over all possible neighboring datasets.

The variance of the Laplace distribution is $2(\Delta/\varepsilon)^2$. Therefore, the magnitude of the added noise will be proportional to the sensitivity and inversely proportional to ε. Decreasing ε flattens out the noise distribution, resulting in noisier estimates. For a given ε, increasing the sensitivity also results in more perturbation.

Global sensitivity The global sensitivity of a function (vector) as defined in 2.10 represents the magnitude by which a single individual's data can change the function in the worst case. It typically depends on N, the size of the input database and is on the order of $O(N^{-1})$ (Hall, 2013). This definition of sensitivity is natural: it gives an upper bound on the amount of perturbation necessary to preserve privacy.

Consider again the Laplace mechanism. The l_1-norm allows decomposition along each vector dimension:

$$\begin{aligned}
dP_X(\mathbf{z}) &\propto \exp\left(-\frac{\varepsilon}{\Delta}\|\mathbf{z}-\boldsymbol{\delta}_X\|_1\right) && (2.11)\\
&\propto \exp\left(\frac{\varepsilon}{\Delta}(|z_1-\delta_1|+\cdots+|z_D-\delta_D|)\right) && (2.12)\\
&\propto \exp\left(\frac{\varepsilon}{\Delta}|z_1-\delta_1|\right)\ldots\exp\left(\frac{\varepsilon}{\Delta}|z_D-\delta_D|\right) && (2.13)\\
&= \left(\text{Laplace}\left(\delta_1,\frac{\Delta}{\varepsilon}\right),\ldots,\text{Laplace}\left(\delta_D,\frac{\Delta}{\varepsilon}\right)\right) && (2.14)
\end{aligned}$$

That is to say, the Laplace mechanism can be applied by sampling each direction independently from a Laplace distribution with scale the sensitivity over the privacy parameter.

The sensitivity is also defined with respect to the l_1-norm. It too can be decomposed along each dimension:

$$\max_{X \sim X'} \|\boldsymbol{\delta}_X - \boldsymbol{\delta}_{X'}\|_1 = \max \|(\delta_1, \delta_2, \ldots, \delta_D) - (\delta_1^{'}, \delta_2^{'}, \ldots, \delta_D^{'})\|_1 \tag{2.15}$$

$$= \max \|(\delta_1 - \delta_1^{'}, \delta_2 - \delta_2^{'}, \ldots, \delta_D - \delta_D^{'})\|_1 \tag{2.16}$$

$$= \max(|\delta_1 - \delta_1^{'}| + |\delta_2 - \delta_2^{'}| + \cdots + |\delta_D - \delta_D^{'}|) \tag{2.17}$$

$$= \max|\delta_1 - \delta_1^{'}| + \max|\delta_2 - \delta_2^{'}| + \cdots + \max|\delta_D - \delta_D^{'}|, \tag{2.18}$$

where we have denoted $\boldsymbol{\delta}_{X'} = (\delta_1^{'}, \ldots, \delta_D^{'})$. Thus, $\Delta = \Delta_1 + \Delta_2 + \cdots + \Delta_D$, where Δ_i is the sensitivity along dimension i. Let us decompose the privacy budget as well, such that $\varepsilon = \varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_D$. What would happen if each direction was sampled independently from a Laplace distribution with the respective sensitivity and privacy budget only, i.e. $\left(\text{Laplace}\left(\delta_1, \frac{\Delta_1}{\varepsilon_1}\right), \ldots, \text{Laplace}\left(\delta_D, \frac{\Delta_D}{\varepsilon_D}\right)\right)$? Direction 1 preserves ε_1-differential privacy, direction 2 preserves ε_2-differential privacy and so on. The joint probability of $\mathbf{z}$ is the product of the probabilities in each direction. The ε-differential privacy guarantee is preserved:

$$\frac{P_X(z_1, \ldots, z_D)}{P_{X'}(z_1, \ldots, z_D)} = \frac{P_X(z_1)}{P_{X'}(z_1)} \times \cdots \times \frac{P_X(z_D)}{P_{X'}(z_D)} \leq \exp(\varepsilon_1) \ldots \exp(\varepsilon_D) = \exp(\varepsilon_1 + \cdots + \varepsilon_D). \tag{2.19}$$

The above holds for any split of the sensitivity or the total privacy budget. If, however, $\Delta_i = \Delta/D$ and $\varepsilon_i = \varepsilon/D$, then

$$\left(\text{Lap}\left(\delta_1, \frac{\Delta_1}{\varepsilon_1}\right), \ldots, \text{Lap}\left(\delta_D, \frac{\Delta_D}{\varepsilon_D}\right)\right) = \left(\text{Lap}\left(\delta_1, \frac{\Delta}{\varepsilon}\right), \ldots, \text{Lap}\left(\delta_D, \frac{\Delta}{\varepsilon}\right)\right). \tag{2.20}$$

To summarize, the differential privacy guarantee permits both different sensitivities and different privacy budgets for each dimension. The Laplace mechanism is applied to each dimension independently. We now continue with the proof of the Laplace mechanism's differential privacy guarantee.

Proof. The proof of the Laplace mechanism's ε-differential privacy is from (Dwork, 2008), also replicated in (Hall, 2013), and relies solely on the triangle inequality on norms (listed below).

$$\frac{dP_X(\mathbf{z})}{dP_{X'}(\mathbf{z})} = \exp\left(\frac{\varepsilon}{\Delta}(\|\boldsymbol{\delta}_{x'} - \mathbf{z}\|_1 - \|\boldsymbol{\delta}_x - \mathbf{z}\|_1)\right) \tag{2.21}$$

$$\leq \exp\left(\frac{\varepsilon}{\Delta}\|\boldsymbol{\delta}_X - \boldsymbol{\delta}_{X'}\|_1\right) \tag{2.22}$$

$$= \exp(\varepsilon). \tag{2.23}$$

Finally, bounding the densities allows us to bound the probabilities as well:

$$P_X(\mathbf{z}) = \int_{\mathcal{Z}} dP_X \leq \int_{\mathcal{Z}} e^{\varepsilon} dP_{X'} = e^{\varepsilon} P_{X'}(\mathbf{z}).$$

We can prove similarly that

$$\frac{P_{X'}(\mathbf{z})}{P_X(\mathbf{z})} \leq e^{\varepsilon}.$$

This completes the proof. □

Using different norms Following are stated three facts, which will motivate a more general version of the Laplace mechanism which replaces the l_1-norm.

1. **Triangle inequality** Also called Minkowski's inequality for general norms, $\|x+y\|_p \leq \|x\|_p + \|y\|_p \;\; \forall p \in \mathbb{R},$ such that $p \geq 1$.

2. **Order on norms** Using Minkowski's inequality it is further possible to prove that the l_1-norm is the largest, i.e. $\|a\|_p = (\sum_{i=0}^{n} |a_i|^p)^{1/p} \leq (\sum_{i=0}^{n-1} |a_i|^p)^{1/p} + |a_n^p|^{1/p} \leq \cdots \leq \sum_{i=0}^{n} |a_i| = \|a\|_1$. More generally, $\|*\|_q \leq \|*\|_p$ whenever $p \leq q$, i.e. the Euclidean norm of a vector is smaller than the Manhattan and so on.

The proof of the Laplace mechanism depends solely on the triangle inequality of norms. Making use of the more general Minkowski's inequality, it is easy to see that an analogue of the Laplace mechanism is possible that replaces the l_1-norm with a p-norm. The resulting mechanism samples a noise vector from the following distribution:

$$dP_X(\mathbf{z}) \propto \exp\left(-\frac{\varepsilon}{\Delta_p}\|\mathbf{z} - \boldsymbol{\delta}_X\|_p\right), \tag{2.24}$$

where Δ_p is the sensitivity defined with respect to the p-norm. The mechanism is still ε-differentially private, even if the sensitivity is defined with respect to the l_1-norm:

$$\begin{aligned}
\frac{dP_X(\mathbf{z})}{dP'_X(\mathbf{z})} &= \exp\left(\frac{\varepsilon}{\Delta}(\|\boldsymbol{\delta}'_x - \mathbf{z}\|_p - \|\boldsymbol{\delta}_x - \mathbf{z}\|_p)\right) \\
&\leq \exp\left(\frac{\varepsilon}{\Delta}\|\boldsymbol{\delta}_X - \boldsymbol{\delta}'_X\|_p\right) \\
&\leq \exp\left(\frac{\varepsilon}{\Delta}\|\boldsymbol{\delta}_X - \boldsymbol{\delta}'_X\|_1\right) \\
&= \exp(\varepsilon).
\end{aligned}$$

Sampling from the distribution in 2.24 is more difficult compared with the Laplace mechanism. Here, the distribution no longer decomposes along each direction. Having said that, it is possible to sample from 2.24 algorithmically by sampling the direction of $\mathbf{z}$ uniformly at random, and sampling the norm of $\mathbf{z}$ from $\Gamma(D, \frac{\Delta_p}{\varepsilon})$ (Hall, 2013).

Let us compare the variance of the noise vector sampled from 2.24 first using the l_1-norm and then the l_2-norm for a D dimensional problem. Let the sensitivity be defined respecting the corresponding norm, i.e. Δ_{l_1} be the sensitivity with respect to the l_1-norm and Δ_{l_2} be the sensitivity with respect to the l_2-norm. The l_1-norm allows us to sample each dimension independently, and so the norm of the resulting vector will be the sum of the norms in each dimension. The variance of a sum of independent variables is the sum of their variances:

$$\text{Var}[|\mathbf{z}|] = \text{Var}[|z_1|] + ... + \text{Var}[|z_D|]. \tag{2.25}$$

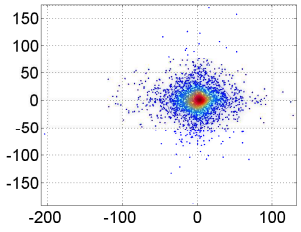
The norm along each direction is exponentially distributed with scale Δ_{l_1}/ε. Therefore,

$$\text{Var}[|\mathbf{z}|] = \sum_d \left(\frac{\Delta_{l_1}}{\varepsilon}\right)^2 = D\left(\frac{\Delta_{l_1}}{\varepsilon}\right)^2 \tag{2.26}$$

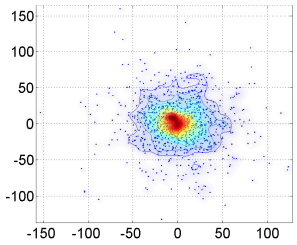
For the l_2-norm, the norm of the vector is Gamma-distributed with scale $\frac{\Delta_{l_2}}{\varepsilon}$ and has variance of $D\left(\frac{\Delta_{l_2}}{\varepsilon}\right)^2$, which only differs from Equation 2.26 in the sensitivity. Even though the variance of the norm of the noise vector reduces if the sensitivity is defined with a form different from l_1, the total amount of perturbation resulting from privacy does not necessarily.

Consider Figure 2.1 which shows samples from the Laplace mechanism, as well as the more general mechanism with different norms, where the sensitivity along each dimension is 1. Figure 2.1a shows the Laplace mechanism with scale $\Delta_{l_1}/\varepsilon = 2/\varepsilon$; Figures 2.1b and 2.1c show samples from the more general mechanism with norms 2 and 10, respectively, i.e. scales $\Delta_{l_2}/\varepsilon = \sqrt{2}/\varepsilon$ and $\Delta_{10}/\varepsilon = \sqrt[10]{2}/\varepsilon$. The distribution in 2.1a is diamond-shaped, while the other two are closer to a circle. When the norm is changed, so is the shape of the distribution of the magnitude of the noise vector. In our 2-dimensional example, the more general norms allow big changes in all directions equally, whereas the l_1-norm allows big changes to the horizontal and vertical dimension only and keeps the diagonal changes relatively small. In terms of the final performance of the private mechanism, it may be better to allow big perturbations in only one dimension, thus eliminating possible benefits from using general norms.

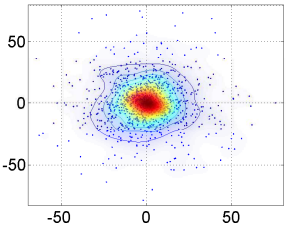
Having said that, one reason for using a p-norm is that the l_1-norm is not everywhere differentiable. In particular, in scenarios where the private mechanism optimizes an objective, it might be necessary or beneficial to add a noise vector prior to optimization. Optimization methods are often gradient-based and require that the objective function is differentiable. Thus it is necessary to avoid the l_1 norm. An example application of



(a) Samples from Laplace(0, $2/\varepsilon$).



(b) Samples from 2.24 with mean 0 and scale $\sqrt{2}/\varepsilon$.



(c) Samples from 2.24 with mean 0 and scale $\sqrt[10]{2}/\varepsilon$.

Figure 2.1: 2000 samples from each distribution together with the probability density as estimated on a grid set up by the range of the data. The norms used are l_1, l_2, and p-norm with $p = 10$, respectively.

the l_2-norm appears in (Chaudhuri et al., 2011). More details on optimization with the l_2-norm are available in chapter 4.

Accuracy of the Laplace mechanism The following theorem gives the accuracy bounds with respect to the sample complexity for the Laplace mechanism and is taken from (Dwork and Roth, 2014). First, we state a general fact about the Laplace distribution, which is necessary for the proof of Theorem 2.2.2. If $\mathbf{z} \sim \text{Laplace}\left(0, \frac{\Delta}{\varepsilon}\right)$,

$$P(\mathbf{z} \geq tb) = \exp(-t). \tag{2.27}$$

Theorem 2.2.2. For a counting query $f\colon \mathcal{X} \to \mathbb{R}^D$ and an output of the Laplace mechanism $\mathbf{z}$, where $\mathbf{z} \sim \text{Laplace}\left(\boldsymbol{\delta}_X, \frac{\Delta}{\varepsilon}\right)$:

$$P\left(\|\delta_X - \mathbf{z}\|_\infty \geq \left(\log \frac{D}{p}\right)\left(\frac{\Delta}{\varepsilon}\right)\right) \leq p. \tag{2.28}$$

where $\mathcal{X}$ is the set of datasets; D is the number of queries that we want answered; $\|\mathbf{x}\|_\infty$ is the max norm defined as the maximum of the absolute values of $\mathbf{x}$'s components; Δ is the sensitivity of the mechanism; and $p \in (0, 1]$ is a probability.

As an example application of Theorem 2.2.2 consider counting the frequency of 1000 names in a dataset of participants in an annual census (Dwork and Roth, 2014). This is a counting query, where the response is $\mathbb{R}^{1000}$ and the sensitivity is 1. If $\varepsilon = 1$ and $p = 0.05$, then with probability 95%, no estimate will be off by more than an additive error of $\log \frac{1000}{0.05} \approx 10$. In our example this is a relatively good guarantee: our estimate will be relatively good because we have a large number of samples (the participants in the census).

Geometric mechanism The Laplace mechanism is well-suited to real-valued outputs. To use the Laplace mechanism when the output of the private mechanism is discrete, it is necessary to round the output of the Laplace mechanism to an integer. Rounding preserves ε-differential privacy, because it is a post-processing operation (see Theorem 2.3.1).

One might wonder if a private mechanism is available which is more privacy efficient in cases where the output of the wanted mechanism is discrete. Indeed, an approach called the Geometric mechanism exists, which samples the output of the private mechanism from a symmetric Geometric distribution. First proposed by (Ghosh et al., 2012) in the context of fixed count queries it has been shown to be expected-loss-minimizing for all possible users, regardless of their background knowledge, subject to the differential privacy constraint.

Having said that it is unlikely that the Geometric mechanism outperforms the Laplace mechanism in practice. According to (Ghosh et al., 2012), the Laplace mechanism is itself approximately universally utility maximizing for most interesting privacy levels, for the authors' definition of utility. In fact, for the usual values of ε, the Laplace mechanism and the Geometric mechanism have nearly identical utility guarantees.

This observation is further confirmed by (Geng and Viswanath, 2013). Their work has found that for histogram release, the optimal solution is a symmetric staircase mechanism, which, for queries of sensitivity 1, is identical to the Geometric mechanism. They have further shown that in practice, the Laplace mechanism is not significantly outperformed by the staircase mechanism.

2.2.2 Exponential Mechanism

The exponential mechanism (McSherry and Talwar, 2007) is a generalization of the Laplace mechanism useful in situations where the wanted output is discrete or categorical. All possible differentially private algorithms could be phrased as the exponential mechanism, because it allows the possible outputs of the mechanism to be rated by any criterion which seems reasonable in the particular domain.

The exponential mechanism like the Laplace mechanism defines a distribution over the range of the function to be computed privately. It achieves this by the use of a quality function: a function which judges the usefulness of an output to the user of the private mechanism. The distribution arising from the exponential mechanism is intuitively fitting. The true output is the most likely, and other outputs get exponentially less likely

with their decreasing quality. Taking the quality function to be the negative of the cost function, the exponential mechanism effectively samples a Boltzmann distribution. Listed below is the Boltzmann probability density function with energy parameter $E(\mathbf{z})$ and a temperature parameter β:

$$dP_X(\mathbf{z}) = \frac{1}{Z}\exp\left(-\beta E(\mathbf{z})\right), \tag{2.29}$$

where Z is the normalization constant.

Distance Metric vs Fitness score In the differential privacy literature, the function which specifies the quality of possible outputs is often referred to as the quality score function or fitness score function. The energy parameter of the Boltzmann distribution is considered a cost, so the quality function is the negative energy, $-E(z)$. This function shows how good the output of the private mechanism is compared to the truth and increases as the "goodness" of the output increases.

To be in accordance with the standard definition of a Boltzmann distribution, this text will prefer the term distance metric. It is a cost function that shows how bad the output of the private mechanism is compared to the true output. The distance metric function increases as the "badness" of the output increases.

Theorem 2.2.3. The Exponential mechanism, 2.30, achieves ε-differential privacy.

$$dP_X(\mathbf{z}) = \frac{1}{Z(X)}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X)\right) \tag{2.30}$$

The idea behind the mechanism is to make high quality outputs exponentially more likely at a rate that depends on the sensitivity of the quality score and the privacy parameter. Intuitively, the mechanism can be thought of a noisy-max picking a noisy "best" output.

The exponential mechanism looks similar to the Laplace mechanism (more on this later), but has a 2 in the denominator of the exponent. The additional 2 results from the ratio of normalization constants $Z(X)/Z(X')$ contributing another factor of $\exp(\varepsilon/2)$ to the ratio of $dP_X(\mathbf{z})/dP'_X(\mathbf{z})$. The complete proof of the exponential mechanism provides more details. This is shown in the discrete case, but the continuous case is similar.

Proof. The exponential mechanism creates a probability distribution on the output space with probability mass function

$$dP_X(\mathbf{z}) = \frac{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X)\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)}. \tag{2.31}$$

To prove differential privacy we need

$$e^{-\varepsilon} \leq \frac{P_X(\mathbf{z})}{P_{X'}(\mathbf{z})} \leq e^{\varepsilon}. \tag{2.32}$$

We start by bounding the ratio of the probability mass functions:

$$\frac{dP_X(\mathbf{z})}{dP_{X'}(\mathbf{z})} = \frac{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X)\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)} \frac{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X')\right)}{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X')\right)} \tag{2.33}$$

$$= \frac{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X)\right)}{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X')\right)} \frac{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X')\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)}. \tag{2.34}$$

Starting with the first multiplier

$$\frac{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X)\right)}{\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z},X')\right)} = \exp\left(\frac{\varepsilon}{2\Delta}(d(\mathbf{z},X') - d(\mathbf{z},X))\right) \leq \exp\left(\frac{\varepsilon}{2\Delta}\Delta\right) = \exp\left(\frac{\varepsilon}{2}\right), \tag{2.35}$$

since the sensitivity is the maximum change in the distance function over any two neighboring datasets $X \sim X'$. In addition, in the second multiplier we can substitute for $d(\mathbf{z}',X)+\Delta$ for $d(\mathbf{z}',X')$, since $d(\mathbf{z}',X') \leq d(\mathbf{z}',X)+\Delta$

$$\frac{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X')\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)} \leq \frac{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}(d(\mathbf{z}',X)+\Delta)\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)} \tag{2.36}$$

$$= \frac{\exp\left(-\frac{\varepsilon}{2}\right)\sum_{\mathbf{z}'}\exp\left(\frac{1}{\Delta}d(\mathbf{z}',X)\right)}{\sum_{\mathbf{z}'}\exp\left(-\frac{\varepsilon}{2\Delta}d(\mathbf{z}',X)\right)} \tag{2.37}$$

$$= \exp\left(\frac{\varepsilon}{2}\right) \tag{2.38}$$

Thus,

$$\frac{dP_X(\mathbf{z})}{dP_{X'}(\mathbf{z})} \leq e^{\varepsilon/2}e^{\varepsilon/2} = e^{\varepsilon} \tag{2.39}$$

Integrating, to get probabilities, we obtain

$$P_X(\mathbf{z}\in\mathcal{Z}) = \int_{\mathcal{Z}} dP_X(\mathbf{z}) \leq \int_{\mathcal{Z}} e^{\varepsilon} dP_{X'}(\mathbf{z}) = e^{\varepsilon}\int_{\mathcal{Z}} dP_{X'}(\mathbf{z}) = e^{\varepsilon}P_{X'}(\mathbf{z}\in\mathcal{Z}), \tag{2.40}$$

which completes the proof. □

The quality score function is user-defined and could be any function which embodies the usefulness of outputs, such that the final distribution is not too flat. Otherwise, bad outputs would be too probable and the mechanism will suffer in performance compared with its non-private counterpart. Indeed, consider the exponential mechanism with a quality score the distance from the true output.

The Laplace Mechanism is a special case of the Exponential Mechanism When the output of an algorithm is $\mathbb{R}^D$ with the appropriate choice of norm and score function,

the exponential mechanism becomes the Laplace mechanism. Indeed, consider the exponential mechanism with distance metric the distance from the true answer multiplied by 2:

$$d(\mathbf{z}, X) = 2\|\boldsymbol{\delta}_{\boldsymbol{X}} - \mathbf{z}\|_1. \tag{2.41}$$

The sensitivity of this is also scaled by 2 and so the 2 disappears. Now the Laplace mechanism and the Exponential Mechanism have the same form.

Private MLE We are interested in reporting the parameters of a machine learning model in a way that preserves the privacy of the dataset. Maximum likelihood is one way to fit the model parameters, which embodies the intuition that the parameters which are made the most likely by the data should be chosen. It is thus very natural to consider the likelihood function as the quality score function of the exponential mechanism; even further, because of the exponent in the exponential mechanism, it makes sense to consider the log-likelihood instead.

If we use the negative log likelihood ($-\log \mathcal{L}(\boldsymbol{\theta})$) as the distance function, we could think of $\mathbf{z}$ as ranging over the possible model parameters $\boldsymbol{\theta}$. Here $\varepsilon = 1/T$ (the inverse temperature). When ε goes to 0 (through positive values) we tend to pick a random sample $\mathbf{z}$ and report that as the output of the privacy mechanism. When ε goes to infinity, $P_X(\mathbf{z})$ will pick the $\boldsymbol{\theta}$ for which the likelihood is the highest, i.e. the MLE. We can think of the exponential mechanism with a fixed $\varepsilon < 1$ as sampling from a more diffuse version of the true posterior with an uninformative prior and thus finding a value close to the true maximum likelihood.

Sensitivity of the NLL In order to use the exponential mechanism, we need to find the sensitivity of the distance function. When we use the NLL as the metric, this means we need to find Δ, such that:

$$\max_{X \sim X'} \|\log \mathcal{L}(\boldsymbol{\theta}|X) - \log \mathcal{L}(\boldsymbol{\theta}|X')\|_1 \leq \Delta. \tag{2.42}$$

However, in general, $\mathcal{L}(\boldsymbol{\theta}) \in [0, \infty)$ for real-valued data and so $\log \mathcal{L}(\boldsymbol{\theta}) \in (-\infty, \infty)$. When data is discrete, $\mathcal{L}(\boldsymbol{\theta}) \in [0, 1]$ for real-valued data and so $\log \mathcal{L}(\boldsymbol{\theta}) \in (-\infty, 0]$. In both cases the sensitivity is unbounded.

Thresholding Because the sensitivity of the likelihood is unbounded, it is impossible to use the NLL as the distance function of the exponential mechanism without any modification. The modification Williams and McSherry (2010) propose is to set the sensitivity to 1, i.e.

$$\Delta = 1.$$

For the negative log likelihood distance function, the above translates as follows:

$$\|\log \mathcal{L}(\boldsymbol{\theta}|X) - \log \mathcal{L}(\boldsymbol{\theta}|X')\|_1 \leq 1 \tag{2.43}$$

$$\text{or} \tag{2.44}$$

$$-1 \leq \log \mathcal{L}(\boldsymbol{\theta}|X) - \log \mathcal{L}(\boldsymbol{\theta}|X') \leq 1. \tag{2.45}$$

The likelihood of the parameters is the product of the probabilities of individual examples, so the log likelihood is a sum of the log likelihoods of each example. The two datasets X and X' are neighbouring. Let the only extra example in X' be called $(\mathbf{x}, y)$. Now the sensitivity bound becomes:

$$-1 \leq \sum_{i}^{N} \log p(\mathbf{x_i}, y|\boldsymbol{\theta}) - \sum_{i}^{N+1} \log p(\mathbf{x'_i}, y'|\boldsymbol{\theta}) \leq 1, \tag{2.46}$$

$$-1 \leq \log p(\mathbf{x}, y|\boldsymbol{\theta}) \leq 1, \tag{2.47}$$

which states that the magnitude of the log likelihood of every sample is bounded by 1. Exponentiating,

$$e^{-1} \leq p(\mathbf{x}, y|\boldsymbol{\theta}) \leq e. \tag{2.48}$$

To summarize, not all distance functions have bounded sensitivity. When confronted with unboundedness, one solution for distance functions which can be factorized is to bound each factor. McSherry and Talwar (2007) refers to this procedure as clamping.

The negative log likelihood makes a natural distance function to be used with the exponential mechanism. As a quality function, the likelihood has to be clamped, meaning that certain inputs get 0 probability. For distributions with thicker tails, the clamping procedure results in significant loss of information and good results are not guaranteed.

2.3 Properties

Differential privacy has a number of attractive properties. Apart from being a formal, provable guarantee, it is the only formulation which allows for the privacy cost to be explicit. Furthermore, it has nice composition properties, which allow datasets with different privacy restrictions to be computed on and the resulting algorithms to be combined. Applications of these theorems are seen throughout this work, but a few basic examples can be referred to in Section 2.3.5. For the proofs, please refer to (Dwork and Roth, 2014).

2.3.1 Invariance to Post Processing

The following theorem states that the output of a differentially private mechanism can be processed further while keeping the ε-differential privacy guarantee. This property of all differentially private mechanisms is useful, because a mechanism's output can be used for purposes beyond the original intent of the private mechanism.

Theorem 2.3.1. Let $\mathcal{P} = \{P_X : X \in \mathcal{X}\}$ be an ε-differentially private algorithm and $f\colon \mathcal{A} \to \mathcal{A}'$. Then $f \circ \mathcal{P}\colon \mathcal{X} \to \mathcal{A}'$ is ε-differentially private.

2.3.2 Sequential Composition

The following theorem applies whenever a dataset is processed multiple times by private algorithms.

Theorem 2.3.2. Let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be n independent mechanisms whose privacy guarantees are $\varepsilon_1, \ldots, \varepsilon_n$-differential privacy, respectively. Then any function g of them: $g(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ is $(\sum_{i=1}^{n} \varepsilon_i)$-differentially private.

2.3.3 Parallel Composition

The following theorem applies whenever disjoint sets of data are input into a differentially private algorithm.

Theorem 2.3.3. Let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be n independent mechanisms whose privacy guarantees are $\varepsilon_1, \ldots, \varepsilon_n$-differential privacy, respectively. If the input datasets to these mechanisms are disjoint subsets of the private database then any function $g(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ is $(\max_i \varepsilon_i)$-differentially private.

2.3.4 Group Privacy

The definition of differential privacy could be used to protect group privacy, instead of privacy of individual rows. In more detail, an adversary with arbitrary auxiliary information cannot know if c particular participants submitted their information. This can be achieved because if c items change, the ratio of probabilities is bounded by $\exp(\varepsilon c)$ instead of $\exp(\varepsilon)$.

2.3.5 Examples

Example with the total sensitivity We now take a look at releasing a two-dimensional vector $\mathbf{z} = \mathcal{P}(X)$, where each dimension has noise added independently and the scale of the noise is the same. $\mathbf{z}$ was sampled from the Laplace mechanism described above, assuming independent elements. In this, we use the total sensitivity in each direction of the noise.

$$\mathbf{z} = (z_1, z_2)^T \sim (\text{Laplace}(\delta_1, \Delta/\varepsilon), \text{Laplace}(\delta_2, \Delta/\varepsilon)). \tag{2.49}$$

In this case

$$P_X(\mathbf{z}) = P_X(z_1)P_X(z_2). \tag{2.50}$$

The private mechanism $\mathcal{P}$ releasing the vector $\mathbf{z}$ as above is ε-differentially private.

Proof. We have to prove that

$$\frac{P_X(\mathbf{z} \in \mathcal{Z})}{P_{X'}(\mathbf{z} \in \mathcal{Z})} \leq \exp(\varepsilon) \tag{2.51}$$

for a subset $\mathcal{Z}$ of $\mathbb{R}^D$.

Using the triangle inequality we can bound the densities that characterize the private mechanism:

$$\begin{aligned}
\frac{dP_X(\mathbf{z})}{dP_{X'}(\mathbf{z})} &= \exp\Big(\frac{\varepsilon}{\Delta}(|\mathbf{z}-\delta_1| + |\mathbf{z}-\delta_2| - |\mathbf{z}-\delta_1'| - |\mathbf{z}-\delta_2'|)\Big) && (2.52)\\
&\leq \exp\Big(\frac{\varepsilon}{\Delta}(|\delta_1 - \delta_{1'}| + |\delta_2 - \delta_{2'}|)\Big) && (2.53)\\
&= \exp(\varepsilon). && (2.54)
\end{aligned}$$

Now we have to integrate to get probabilities. For any subset $\mathcal{Z}$ of $\mathbb{R}^D$,

$$P_X(\mathbf{z} \in \mathcal{Z}) = \int_{\mathcal{Z}} dP_X(\mathbf{z}) \leq \int_{\mathcal{Z}} e^{\varepsilon} dP_{X'}(\mathbf{z}) = e^{\varepsilon} \int_{\mathcal{Z}} dP_{X'}(\mathbf{z}) = e^{\varepsilon} P_{X'}(\mathbf{z} \in \mathcal{Z}), \tag{2.55}$$

so the ratio of the cdfs is bounded as well. □

Example with different ε in each direction If we assume the privacy parameter is different for each direction, we can again obtain differential privacy. The directions are still independent and we again use the total sensitivity in each direction.

$$\mathbf{z} = (z_1, z_2)^T \sim (\text{Laplace}(\delta_1, \Delta/\varepsilon_1), \text{Laplace}(\delta_2, \Delta/\varepsilon_2)). \tag{2.56}$$

Proof.

$$\frac{dP_X(\mathbf{z})}{dP_{X'}(\mathbf{z})} = \exp\left(\frac{\varepsilon_1}{\Delta}(|\mathbf{z}-\delta_1| - |\mathbf{z}-\delta_1'|) + \frac{\varepsilon_2}{\Delta}(|\mathbf{z}-\delta_2| - |\mathbf{z}-\delta_2'|)\right) \tag{2.57}$$

$$\leq \exp\left(\frac{\varepsilon_1}{\Delta}|\delta_1 - \delta_{1'}| + \frac{\varepsilon_2}{\Delta}|\delta_2 - \delta_{2'}|\right) \tag{2.58}$$

$$\leq \exp\left(\frac{\varepsilon_1}{\Delta}\Delta_1 + \frac{\varepsilon_2}{\Delta}\Delta_2\right) \tag{2.59}$$

$$\leq \exp(\varepsilon), \tag{2.60}$$

where $\varepsilon = \max(\varepsilon_1, \varepsilon_2)$. We complete the proof as before. □

This example shows why ε-differential privacy is not particularly well suited to problems which involve several datasets with different privacy restrictions. Even though we would like to be able to provide different levels of privacy to the different datasets, ε-differential privacy can only provide a final guarantee of ε, effectively ignoring the granularity of the requirements. Based on this intuition, Appendix B provides an initial investigation into the combination of models with different privacy levels.

Example with the partial sensitivities We now look at what happens if we add noise from two different Laplace distributions.

$$\mathbf{z} = (z_1, z_2)^T \sim (\text{Laplace}(\delta_1, \Delta_1/\varepsilon_1), \text{Laplace}(\delta_2, \Delta_2/\varepsilon_2)). \tag{2.61}$$

Two dependent numbers z_1 and z_2 released with guarantees ε_1 and ε_2, respectively, can be released together as follows, if we add noise proportional to the respective dimension:

$$\frac{P_X(z_1, z_2)}{P_{X'}(z_1, z_2)} = \frac{P_X(z_1)}{P_{X'}(z_1)} \times \frac{P_X(z_2)}{P_{X'}(z_2)} \leq \exp(\varepsilon_1)\exp(\varepsilon_2) \leq \exp(\varepsilon_1 + \varepsilon_2), \tag{2.62}$$

where X' is the neighboring dataset.

To sum up, if we add noise to each direction using the sensitivity of this direction only, we have to add up the epsilons. If we use the total sensitivity for each direction, we can just use the maximum ε.

Let's say we are releasing a vector and the total sensitivity of the mechanism is D. Let's also assume that the total budget we have is ε. According to the above, we could choose between

$$\mathbf{z} = (z_1, \ldots, z_D)^T \sim (\text{Laplace}(\delta_1, \Delta_1/\varepsilon_1), \ldots, \text{Laplace}(\delta_D, \Delta_2/\varepsilon_D)) \tag{2.63}$$

and

$$\mathbf{z} = (z_1, \ldots, z_D)^T \sim (\text{Laplace}(\delta_1, \Delta/\varepsilon), \ldots, \text{Laplace}(\delta_D, \Delta/\varepsilon)), \tag{2.64}$$

where $\Delta = \Delta_1 + \cdots + \Delta_D$ and $\varepsilon = \varepsilon_1 + \cdots + \varepsilon_D$.

Consider a scenario where the sensitivity Δ is the number of dimensions and the sensitivity Δ_d in each direction is 1. This is a typical scenario in counting queries, where each dimension of the output of the mechanism in question is a count. In addition, let the total privacy budget be split equally amongst all dimensions, such that $\varepsilon_d = \varepsilon/D$. It is easy to see that the distributions 2.63 and 2.64 will be the same:

$$\frac{\Delta_d}{\varepsilon_d} = \frac{1}{\varepsilon/D} = \frac{D}{\varepsilon} = \frac{\Delta}{\varepsilon}.$$

This fact will prove useful in the design of the private Naive Bayes classifier.

2.4 Private Machine Learning Strategies

The resilience of private mechanism outputs to post-processing allows several alternative approaches for creating differentially private applications.

Input perturbation Consider representing a dataset as a collection of vectors, real-valued or otherwise. The dataset can be protected by adding privacy related noise according to a certain privacy level. Any computation using the protected dataset, private or otherwise, will guarantee the differential privacy of the original dataset (Theorem 2.3.1 on page 19). This is the basis for the input perturbation approach. Probably the most straightforward technique for private computation, it protects the dataset before it does any computation with it.

In more detail, each D-dimensional entry $\mathbf{x}$ in the dataset is obtained by

$$\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{b} \tag{2.65}$$

where $\mathbf{b}$ is a random D-dimensional vector with a Laplace density

$$p(\mathbf{b}) \propto \exp\left(-\frac{\varepsilon}{2\Delta}\|\mathbf{b}\|_1\right), \tag{2.66}$$

where Δ is the maximum change in the l_1-norm of $\mathbf{x}$ due to its substitution with a different vector. Due to the properties of the l_1-norm, each element of $\mathbf{b}$ can be sampled independently from a Laplace distribution scaled by the maximum change in the corresponding direction. Applying 2.65 to all entries in the dataset produces an ε-differentially private approximation to the original dataset.

Output perturbation Analogously to the above method, one can perturb the output of an algorithm to preserve privacy. For example, to produce a private version of the MLE for logistic regression, one would train logistic regression as usual; then sample

a noise vector with the same dimensionality and return the original estimate plus the noise vector. Indeed this is one approach taken by (Chaudhuri et al., 2011) in designing a private logistic regression classifier. For example, if the output of a non-private algorithm is δ, instead of it, we could release

$$\tilde{\delta} = \delta + \mathbf{b}, \tag{2.67}$$

where $\mathbf{b}$ is drawn from the Laplace distribution with zero mean and scale parameter Δ/ε and $\theta_{\mathcal{X}}$ is the output of the non-private equivalent of $\mathcal{P}_{\mathcal{X}}$. Noise is sampled independently in each dimension. Alternatively, we could use a more general p-norm, as discussed before.

Depending on the domain of application, it is possible that the output's sensitivity is unbounded, in which case it has to be thresholded or otherwise bound to a certain interval.

Objective perturbation Naturally, noise can be added at any intermediate step of an algorithm, as long as the sensitivity of the vector reported is used. One example of this is the objective perturbation method, where a noise vector is applied to the objective function prior to optimization, as will be described in great detail in Chapter 3.

Choosing a strategy The choice of strategy is not always easy, but should be guided by the principle of least information (Dwork and Roth, 2014). The principle states that a private mechanism should output the least possible amount of information required to complete a task. Intuitively, this is correct because the less information an output carries, the less it will have to be perturbed to preserve privacy.

Usually, the private mechanism takes in some data and computes some statistic from it. Taking into account the principle of least information (Dwork and Roth, 2014), it is unlikely that perturbing the data itself is the optimal choice. The data can carry more information than necessary to compute the output of interest, and it is usually high-dimensional and of different types. However, input perturbation could be useful in situations where the data is used in more than one ways, or whenever the data analyst wants to use a non-private algorithm.

Following the principle of least information, it is the output perturbation that seems like the optimal strategy and it may well be in many cases. Mechanisms take in a lot of data, make complex computations on it, but often output a vector whose sensitivity is much smaller than that of the intermediate steps.

Previous research (Chaudhuri et al., 2011; Kifer et al., 2012) has suggested that objective perturbation may perform better than output perturbation on "nice data". In

their example, these are datasets for which the loss function is strongly convex in a neighborhood of its minimizer, the objective perturbation has better error guarantees than in the worst-case. Some experiments have shown a slight advantage for objective perturbation, but more experiments are needed to confirm or deny the observable gains in practice.

Sample-and-aggregate framework The sample-and-aggregate framework (Nissim et al., 2007) is a generic technique for designing private algorithms. It is most useful when the function of interest has difficult to bound or analyze global sensitivity, but is believed to be well-behaved in practice. This framework is particularly useful when the function can be accurately estimated on a small sample set.

The sample-and-aggregate method works as follows. First the dataset is partitioned into several blocks of equal size. Next, the function of interest is computed independently on each block exactly, without noise. The intermediate outcomes are then combined via a differentially private aggregation mechanism, typically a mean or a median. The sensitivity of the aggregation function is used, and this is low, since only one of the blocks will be changed as a result of adding one adding one new entry to the original dataset. Some example applications of this framework include (Chaudhuri et al., 2012) and (Czerniak and Zarzycki, 2003).

2.5 Existing Machine Learning Applications

This thesis is concerned with extending the applications of differential privacy to machine learning. Some background on the terminology used in this thesis and in the wider literature is in order.

A machine learning algorithm takes as input a set of samples or examples, called a training set, and builds a model which captures knowledge about the underlying distribution of the data. The learned model can be used to make predictions or decisions about newly seen data.

Often, when discussing the privacy of a dataset, we are concerned with the privacy of the individuals whose data is in the dataset. Usually, there is one entry (sample) per person in a dataset, which is why the literature talks about individuals rather than samples. The set of all possible samples is called the sample space and the dimensionality of the sample space refers to the number of variables (features) each sample has.

Machine learning is divided into two main types: predictive or supervised and

unsupervised (Murphy, 2012). In supervised learning the goal is to learn a mapping from inputs $\boldsymbol{x}$ to outputs y, given a training set of N input-output pairs $X = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. The features of $\boldsymbol{x}$ can be categorical or numerical, and y can be categorical or real-valued. When y is categorical, the problem is known as classification; when y is real-valued, as regression. In unsupervised learning, the training set consists only of inputs $\{(\boldsymbol{x}_n)\}_{n=1}^{N}$ and the goal is to find interesting patterns in the data.

This work will focus on binary classification, as there are established well understood machine learning algorithms which have numerous real-world applications. Before we pursue the goal of practical binary classifiers, we review some of the existing privacy-preserving machine learning literature.

2.5.1 Evaluation criteria

To evaluate the quality of a differentially private model, it is necessary to compare against a "true model". In the literature, the true model is usually taken to be the model learned by a non-private fitting procedure on the training data.

Having established the true model, we have to measure the distance between it and the private model. A number of works (Chaudhuri et al., 2011; 0002 et al., 2012; 0002 and Thakurta, 2013) take the distance between the private and non-private model to be the difference in their test accuracy, i.e. the proportion of correctly predicted labels on previously unseen data. Other works (Dwork and Lei, 2009; Chaudhuri et al., 2012; Lei, 2011) compare the distance between the model parameters learned by the two models. We have used both of these techniques, focusing on the difference in accuracies in experiments with real datasets, and investigating differences in parameter values in diagnosing bad performance.

Having defined a distance measure between the true and the private model, some works, such as (Chaudhuri et al., 2011; Blum et al., 2013), provide convergence rate guarantees in terms of (α, β)-usefulness (Blum et al., 2008). The output of a private mechanism is (α, β)-useful if with probability $1 - \beta$ the difference between it and the true output is less than α. Although a useful notion, in this work we prefer comparisons based on empirical performance, rather than theoretical guarantees.

2.5.2 Differentially private supervised learning

Vaidya et al. (2013) offer a differentially private Naive Bayes classifier for numerical and categorical features. They use the Laplace mechanism and compute the sensitivity

of the mechanism output. We defer further discussion of this paper to Chapter 3.

Differentially private logistic regression classifiers have been developed by (Chaudhuri and Monteleoni, 2008; Chaudhuri et al., 2011) and (Zhang et al., 2012). Chaudhuri and Monteleoni (2008); Chaudhuri et al. (2011) propose two methods for private logistic regression. The first approach is to perturb the model parameters learned via non-private logistic regression. As previously described, this is the output perturbation method, which adds noise from a symmetric distribution. Further details on output perturbation are available in Chapter 4. The second approach presented in (Chaudhuri et al., 2011) is objective perturbation. It adds noise from the same parametric family of distributions, but perturbs the objective function before it is optimized. Detailed explanation of this approach is deferred to Chapter 4. Output perturbation requires fewer assumptions than objective perturbation. Under these assumptions, namely strong convexity, differentiability, and boundedness of the sample space, which allow both techniques to apply, (Chaudhuri et al., 2011) shows that the worst-case theoretical guarantees of the two methods are very similar.

The output and objective perturbation do not scale well with the dimensionality of the data. Kifer et al. (2012) improve the objective perturbation technique by relaxing the privacy guarantee to (ε, δ)-differential privacy. Furthermore, the relaxation allows the use of objective perturbation for convex programs like the Lasso, where the regularizer is the l_1-norm.

Zhang et al. (2012) offer a different perspective on differentially private logistic regression. The authors' approach is still from an objective perturbation viewpoint, however, they manage to avoid the sensitivity analysis on the original objective by a polynomial representation of it. In order for the optimization to succeed, the polynomial needs to have bounded coefficients. For logistic regression, this is achieved by creating a quadratic Taylor expansion of the objective around 0:

$$J(\theta) \approx \theta^T M \theta + \alpha\theta + \beta. \tag{2.68}$$

Differential privacy is then achieved through the perturbation of the coefficients of Equation 2.68. The sensitivity is found to be twice the total sum of the absolute values of the approximation coefficients and noise is sampled from the Laplace distribution with scale the sensitivity over the privacy parameter. Injecting such noise into the coefficients of the objective function may render the objective unbounded. This is corrected by adding a constant regularization term to make

$$J(\theta) \approx \theta^T (M + \lambda\mathbb{I})\theta + \alpha\theta + \beta. \tag{2.69}$$

In contrast to the objective perturbation method of (Chaudhuri et al., 2011), which adds noise only to the linear term, the functional mechanism perturbs the quadratic term as well, totalling in $O(D^2)$ noised terms. The error surface is both shifted and scaled. Experiments with the functional mechanism, detailed in Appendix B, confirmed that it scales less well with the dimensionality than the objective perturbation method.

Various other machine learning algorithms have been explored under the differential privacy constraint. Privacy-preserving support vector machines have been detailed in Rubinstein et al. (2009); Chaudhuri et al. (2011); private treatment of decision trees is presented in (Jagannathan et al., 2012; Friedman and Schuster, 2010); private online learning is discussed in (Guha Thakurta and Smith, 2013; Jain et al., 2011). Further review of differentially private machine learning algorithms is available in (Ji et al., 2014).

2.5.3 Cross-validation

(Chaudhuri and Vinterbo, 2013a) design a differentially-private cross validation procedure for training algorithms and performance measures which obey a certain stability condition. They arrive at a notion called $(\beta_1, \beta_2, \delta)$-stability, which holds if the performance measure does not change very much when one person's private value in the training or validation set changes, when the same random bits are used in the algorithm. Moreover, the proposed method only works on differentiable penalty functions, such as the l_2-penalty (the standard ridge regression cost function).

The algorithm learns several models, each with α_1-differential privacy, for all possible values of the hyper-parameter. Next, one of these outputs is chosen by an α_2-differentially private mechanism using a validation score. The authors use a procedure called *exponential variables*, but the exponential mechanism is similarly applicable. Finally, the chosen hyper-parameter and the training data is used to retrain and get the final output. The total privacy guarantee of their algorithm is $(\alpha_1 + \alpha_2, \delta)$-differential privacy. (Chaudhuri and Vinterbo, 2013a) is extended in (Wang et al., 2009) to apply to any convex penalty function, including elastic-net and lasso penalties.

In this work, cross-validation is performed in a non-private manner to aid experimentation. However, to formally guarantee ε-differential privacy, cross-validation must be done using a valid private mechanism. We suggest using the method described above, but also encourage future work on the subject.

2.6 Useful definitions

Here are listed definitions, occurring frequently in the rest of this work.

Definition 2.6.1. The mutual information of two discrete random variables X and Y is defined as:

$$I(X_d,Y) = \sum_{y\in Y}\sum_{x_d\in X_d} p(x_d,y)\log\left(\frac{p(x_d,y)}{p(x_d)\,p(y)}\right), \tag{2.70}$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively.

In this work the mutual information will be used in the context of feature selection. In particular, we will often compute the mutual information between a feature and a class label. Most of the features we talk about are binary and the problems we look at are two-class. For binary variables the above formula can be rewritten as follows:

$$I(X_d,Y) = \sum_c \left[\theta_{dc}\pi_c \log\frac{\theta_{dc}}{\theta_d} + (1-\theta_{dc})\pi_c \log\frac{1-\theta_{dc}}{1-\theta_d})\right], \tag{2.71}$$

where $\pi_c = p(y=c)$, $\theta_{dc} = p(x_d = 1|y=c)$, and $\theta_d = p(x_d=1) = \sum_c \pi_c\theta_{dc}$. In this work the logarithm in the definition is base 2.

Definition 2.6.2. In experiments we often return to the majority baseline of a dataset. The majority baseline of a set is the proportion of samples in the most frequently occurring class of the dataset.

2.7 Looking ahead

The differentially private mechanisms detailed here have promising theoretical guarantees: they are approximately utility-maximizing, for a certain definition of utility, asymptotically equivalent to their non-private counterparts, and computationally efficient. Although theoretical work is abundant and our understanding of the workings of these private mechanisms is good, more evidence is needed of their practical utility. The few applications of differentially private machine learning algorithms (Erlich and Narayanan, 2013) fail to convincingly make the case for the widespread adoption of differential privacy amongst data science practitioners. More often than not, the argument has been that differential privacy should be used because it is the best privacy guarantee in existence. The goal of this work is to begin to show the possibility of useful results from a differentially private classifier in a wide variety of realistic settings.

In the following chapter, the Laplace mechanism is used to create a private Naive Bayes classifier, which is then evaluated on real-world datasets. The results are encouraging, although on high-dimensionality datasets additional techniques are needed to boost performance.

Chapter 3

Differentially Private Naive Bayes

3.1 Motivation

High-dimensional data is omnipresent in the real world and has inspired considerable research effort in the machine learning community. The challenges arising in the analysis of high-dimensional datasets are even more pronounced in private algorithms: privacy-related perturbation significantly reduces the signal from the data. In settings where sensitive data is already scarce, successful differentially private algorithms need to be extremely statistically efficient.

Naive Bayes classifiers are one of the oldest and simplest to train probabilistic models. Due to the strong independence assumption between features, they scale well with dimensionality. The maximum likelihood estimate for Naive Bayes reduces to counting queries to a dataset. Counting queries are the most straightforward for differential privacy, because they have a constant sensitivity with respect to the number of samples in the dataset. In addition, counting queries do not require any additional computation apart from counting, they comply with the principle of "information minimization" (Dwork and Roth, 2014). Thus, Naive Bayes is the best candidate in the search for an effective, highly-scalable differentially private classifier.

Furthermore, maximum-likelihood training of a Naive Bayes classifier can be done by evaluating a closed-form expression, computed in linear time. Efficiency undoubtedly has practical implications for differentially private algorithms.

This chapter begins with some background on Naive Bayes classifiers. Next, we develop a Naive Bayes classifier, prove its differential privacy guarantee, and evaluate it empirically on several real world datasets. A dataset partitioning strategy is then investigated, as a way to improve scalability with the number of features. Finally, a

private dimensionality reduction technique based on feature selection is developed, which successfully improves the performance of private Naive Bayes classifiers.

3.2 Background

Naive Bayes is a classifier, a machine learning system that given input features $\mathbf{x}$ of dimensionality D, predicts an unknown class label y, usually a categorical variable. It's based on a generative model where the individual features x_d, for each dimension d, are assumed to be conditionally independent given the class label. Even though the independence assumption is often inaccurate, for example in text classification problems, naive Bayes classifiers have been useful in practice due to their scalability. The naive assumption alleviates the curse of dimensionality, since the sample size does not scale exponentially with the number of features. The class conditional density is a product of the one-dimensional densities:

$$p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{d=1}^{D} p(x_d|y=c,\boldsymbol{\theta}_{dc}), \tag{3.1}$$

where $\boldsymbol{\theta}$ are parameters learned from $(\mathbf{x},y)$, the set of example pairs in the training set.

Our investigation will concern itself only with binary features. Real-valued and categorical features are left as future work, but extensions are similar in nature. For binary features:

$$p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{d=1}^{D} Ber(x_d|\theta_{dc}), \tag{3.2}$$

where θ_{dc} is the probability that feature d occurs in class c.

3.2.1 Maximum Likelihood Estimation

One way to estimate the parameters $\boldsymbol{\theta}$ of a model is through maximum likelihood training, i.e. selecting the set of parameters which maximize the likelihood given the data. Naive Bayes is attractive because the maximum likelihood training can be performed by evaluating a closed-form expression in linear time, as opposed to the iterative approach necessary for many other types of classifiers.

In the standard naive Bayes, the likelihood given an example is (Murphy, 2012):

$$p(\mathbf{x},y|\boldsymbol{\theta}) = p(y|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|\boldsymbol{\theta}_d) = \prod_{c} \pi_c^{\mathbb{I}(y=c)} \prod_{d} \prod_{c} p(x_d|\boldsymbol{\theta}_{dc})^{\mathbb{I}(y=c)}. \tag{3.3}$$

Taking the log,

$$\log(p(\mathbf{x}, y|\boldsymbol{\theta})) = \log \pi_c + \sum_d \sum_c \log p(x_d|\boldsymbol{\theta}_{dc})^{\mathbb{I}(y=c)} \tag{3.4}$$

where $p(x_d|\boldsymbol{\theta}_{dc}) = Cat(x_d|\boldsymbol{\theta}_{dc})$ for categorical features or $p(x_d|\boldsymbol{\theta}_{dc}) = Ber(x_d|\theta_{dc})$ for Bernoulli features. Thus the log likelihood of the model parameters given all training examples is:

$$\log \mathcal{L}(\boldsymbol{\theta}) = \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_c N_c \log \pi_c + \sum_d \sum_c \sum_{i:y=c} \log p(x_d^{(i)}|\boldsymbol{\theta}_{dc}) \tag{3.5}$$

$$= \sum_c N_c \log \pi_c + \sum_d \sum_c \sum_{i:y=c} N_{dc} \log \boldsymbol{\theta}_{dc}, \tag{3.6}$$

where all $\boldsymbol{\theta}_{dc}$'s sum to 1 and N_{dc} is the number of times this feature is on in this class.

Having built a class conditional model using the probability of seeing each feature, given the document class, we are ready to estimate the model parameters using maximum likelihood. We only state the results for Bernoulli distributed θ_{dc}s. The maximum likelihood estimate for the class probabilities is $\bar{\pi} = \frac{N_c}{N}$, i.e. the number of documents of class c over the total number of documents; and for the probability of each class containing each feature, $\bar{\theta}_{dc} = \frac{N_{dc}}{N_c}$, i.e. the number of documents in class c that feature d turns up in over the number of documents in class c.

The problem with maximum likelihood training is that it encourages the model parameters to overfit the data (Murphy, 2012). A solution to overfitting is to place a prior probability on the model parameters, which quantifies our beliefs about them before seeing any data. For Naive Bayes the prior is a factored:

$$p(\boldsymbol{\theta}) = p(\pi) \prod_{d=1}^{D} \prod_{c=1}^{C} p(\theta_{dc}), \tag{3.7}$$

where $\pi \sim \text{Dir}(\boldsymbol{\alpha})$ and $\theta_{dc} \sim \text{Beta}(\beta_0, \beta_1)$. Often $\boldsymbol{\alpha} = 1$ and $\boldsymbol{\beta} = 1$, which is referred to as add-one or Laplace smoothing.

Combining the factored prior with the factored likelihood gives a factored posterior of the model parameters given the data. To compute the posterior probability of the model parameters given the data, we only need to update the prior counts with the empirical counts form the likelihood. Alternatively, we can think of adding a prior as adding fake data points to the training data and then counting. In particular,

$$\bar{\theta}_{dc} = \frac{N_{dc} + \beta_1}{N_c + \beta_0 + \beta_1} \tag{3.8}$$

$$\bar{\pi}_c = \frac{N_c + \alpha_c}{N + \alpha_0}, \tag{3.9}$$

where $\alpha_0 = \sum_c \alpha_c$ and α's and β's are fake counts. In this work we use add-one smoothing and talk about pseudo counts.

When predicting the class of a new sample $\mathbf{x}^*$, we are interested in the posterior probability of a class given the training data, or the posterior predictive density. In the case of Naive Bayes this can be obtained by

$$p(y = c|\mathbf{x}^*, X) \propto \bar{\pi}_c \prod_d (\bar{\theta}_{dc})^{\mathbb{I}(x_d=1)} (1 - \bar{\theta}_{dc})^{\mathbb{I}(x_d=0)}, \tag{3.10}$$

where $\bar{\theta}_{dc}$ and $\bar{pi}_c$ are estimated as above. We can then select the class c which has the highest posterior probability. When all we want to infer are class labels we do not need to compute the normalization constant, since it is the same for all classes.

3.2.2 Designing private Naive Bayes classifiers

To develop a private Naive Bayes classifier, we go through the usual steps involved in private mechanism creation. First, the most appropriate mechanism is chosen, depending on the type of the output - Laplace for numeric and the Exponential Mechanism for categorical, or whenever more flexibility is required in the quality score function definition.

Next, a metric on the output space of the algorithm has to be chosen, which specifies how good a particular output is, given each input to the algorithm. Importantly, this needs to be a low-sensitivity function; otherwise accuracy will suffer as result of too much noise.

All private mechanisms sample from a distribution on this output space which depends on the sensitivity of the metric function and the privacy parameter. Depending on whether the application permits it, several values for the privacy parameter should be experimented with, until a satisfactory trade-off is found between the performance of the private algorithm and the privacy level.

The final step of the design of private Naive Bayes is the choice of perturbation strategy. Input perturbation, i.e. adding noise to the data itself, has been shown (Chaudhuri et al., 2011) to often perform worse than output perturbation. As far as Naive Bayes is considered, the objective perturbation method is ill-fitting: the model parameters that maximize the likelihood can be found analytically and there is no need to optimize a noisy objective function. Therefore the output perturbation is the method of choice for Naive Bayes classifiers.

The first approach to designing a private Naive Bayes classifier is to use the Laplace mechanism and Manhattan distance metric. An alternative approach is to consider

the negative (log) likelihood function as the distance metric. A metric based on the likelihood is intuitively appropriate, since it is a measure of the "goodness" of the model parameters and shows how well these parameters explain the data at hand. We detail the first approach and leave the second for future work.

3.3 Private Bernoulli Naive Bayes using the Laplace Mechanism

This section details the use of the Laplace mechanism to release a noisy maximum a posteriori estimate of a private naive Bayes classifier.

Distance metric We quantify the usefulness of possible outputs of the private mechanism by measuring the Manhattan distance between them and the true (best) possible output. Measuring the quality of the noisy output parameters, this metric is simply the l_1-distance between the true parameter vector and the noisy released parameter vector:

$$d(X, \tilde{\boldsymbol{\theta}}) = \|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|_1. \tag{3.11}$$

3.3.1 Output

Setup and notation The standard Naive Bayes algorithm returns $O(DC)$ parameters, linear in the dimensionality of the data. The private Naive Bayes mechanism described here will release a noisy estimate of the these. We denote the noisy parameters by $\tilde{\theta}_{dc}$, and interpret them as the noisy proportions of times each feature is turned on in each class. A noisy vector of class probabilities, $\tilde{\boldsymbol{\pi}}$, is also released. Altogether,

$$\tilde{\boldsymbol{\pi}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_C)^T \tag{3.12}$$

and

$$\tilde{\boldsymbol{\theta}} = \begin{bmatrix} \tilde{\theta}_{11}^{(1)} & \ldots & \tilde{\theta}_{D1}^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{\theta}_{1C}^{(1)} & \ldots & \tilde{\theta}_{DC}^{(1)} \end{bmatrix}.$$

For brevity, we denote with $\tilde{\boldsymbol{\theta}}$ the vector that contains all of the above, i.e. $\tilde{\boldsymbol{\theta}} = (\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\pi}})$. In the non-private setting, the MLE parameters are $\theta_{dc} = \frac{N_{dc}}{N_c}$ and $\pi_c = \frac{N_c}{N}$. In the private setting our differentially private algorithms will return noisy counts. Noisy estimates of the parameters computed from these noisy parameters will satisfy the same differential

privacy guarantee as the algorithm releasing the counts (Theorem 2.3.1., p16). The noisy counts are denoted as follows:

$$\tilde{\boldsymbol{M}} = (\tilde{N}_1, \ldots, \tilde{N}_C)^T \tag{3.13}$$

and

$$\tilde{\boldsymbol{N}} = \begin{bmatrix} \tilde{N}_{11}^{(1)} & \ldots & \tilde{N}_{D1}^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{N}_{1C}^{(1)} & \ldots & \tilde{N}_{DC}^{(1)} \end{bmatrix}.$$

Sensitivity When analyzing a private mechanism it is always necessary to quantify its sensitivity to changes in the input. Sensitivity is defined as the maximum change in the output of a an algorithm as a result of *adding* a new example in the dataset (see Definitions 2.1.1 and 2.10, p. 7 and p. 9, respectively).

With respect to private Naive Bayes classification, computing the sensitivity means quantifying the maximum change in the released model parameters given the addition of one new training sample. The distance metric for the private Naive Bayes mechanism is the Manhattan distance.

With the addition of one new example, each column of $\tilde{\boldsymbol{N}}$ will increase by at most 1, since the new example belongs to only one of the classes. In total, $\tilde{\boldsymbol{N}}$ will increase by at most D in the 1-norm. $\tilde{\boldsymbol{M}}$ will increase by at most 1, since we've added an example belonging to only one of the N_c's. Therefore, $(\tilde{\boldsymbol{N}}, \tilde{\boldsymbol{M}})$'s 1-norm will change by at most $D+1$, making the sensitivity of the Manhattan distance metric on this output space $D+1$.

Releasing off-counts Consider releasing a vector $(\boldsymbol{N}^{(1)}, \boldsymbol{N}^{(0)})$ which contains the number of times a feature is on as well as the number of times a feature is off, respectively, for each class, $2DC$ parameters in total.

$$\tilde{\boldsymbol{N}} = \begin{bmatrix} \tilde{N}_{11}^{(1)} & \ldots & \tilde{N}_{D1}^{(1)}, & \tilde{N}_{11}^{(0)} & \ldots & \tilde{N}_{D1}^{(0)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{N}_{1C}^{(1)} & \ldots & \tilde{N}_{DC}^{(1)}, & \tilde{N}_{1C}^{(0)} & \ldots & \tilde{N}_{DC}^{(0)} \end{bmatrix}.$$

Improved sensitivity Consider again the change in sensitivity due to the addition of one example. In the matrix above, on each row we have the counts for one class. The first half of the row has the on-counts and the second - the off-counts. Only one row in this matrix will change as a result of a new example. Moreover, the row can change by at most D, since either an on- or an off-count will be updated. Thus the maximum change in the value of the distance metric as a result of adding a new sample is D. The

difference with the previous sensitivity is negligible for large D, but might provide for better results in very low-dimensional problems.

3.3.2 Algorithm

Following the Laplace mechanism, noise is sampled from the Laplace distribution

$$dP_X(\tilde{\boldsymbol{N}}) \propto \exp\left(-\frac{\varepsilon}{D}\|\tilde{\boldsymbol{N}}\|_1\right) \tag{3.14}$$

and added to each direction of $\tilde{\boldsymbol{N}}$ independently. The scale of the noise added to each dimension is $\frac{D}{\varepsilon}$. The algorithm is presented in 1.

To get an appreciation for the magnitude of the noise, the standard deviation of this Laplace distribution for $\varepsilon = 0.1$ and $D = 600$ is $\sqrt{2}\frac{D}{\varepsilon} \approx 8500$; for $\varepsilon = 1$ and $D = 600$ it's ≈ 850. The algorithm reports counts, i.e. the number of times each feature is on in each class. This means the output of the algorithm is bounded by the number of samples available to it. For good accuracy, of the size of the dataset is of the order of 8500 or less, the results of the private mechanism will be completely obliterated by the noise. Thus, for $\varepsilon = 0.1$, the size of the dataset should be much bigger than 8500. Clearly, a significant amount of data will be necessary to achieve satisfactory results with a high level of privacy for a dataset with many features.

Algorithm 1 Private Bernoulli Naive Bayes MLE based on the Laplace mechanism

procedure PRIVATENBFIT1($X, y, pseudoCount, \varepsilon$) ▷ The design matrix X, the vector of target values y, the effective sample size of the prior, the privacy parameter ε

$\Delta = D$

$t_{lower} = -\frac{\Delta}{\varepsilon}\log(2(1-0.65))$

for $c = 1 : C$ **do** ▷ For each class

$Xc =$ all samples in class c

$\boldsymbol{N}^{(1)} =$ the number of times each feature is turned on in c

$\tilde{\boldsymbol{N}}^{(1)} = \max(\boldsymbol{N}^{(1)} + Laplace(0, \Delta/\varepsilon),\ t_{lower})$

$\boldsymbol{N}^{(0)} =$ the number of times each feature is turn off in c

$\tilde{\boldsymbol{N}}^{(0)} = \max(\boldsymbol{N}^{(0)} + Laplace(0, \Delta/\varepsilon),\ t_{lower})$

$\tilde{\boldsymbol{\theta}}(c,:) = [\frac{\tilde{\boldsymbol{N}}^{(1)} + pseudoCount}{\tilde{\boldsymbol{N}}^{(1)} + \tilde{\boldsymbol{N}}^{(0)} + 2pseudoCount}, \frac{\boldsymbol{\theta}^{(0)} + pseudoCount}{\tilde{\boldsymbol{N}}^{(1)} + \tilde{\boldsymbol{N}}^{(0)} + 2pseudoCount}]$

Thresholding to ensure consistency Since the Laplace is a double-sided distribution, it is possible that the counts become negative. This is problematic because we

estimate probabilities using counts and probabilities need to be non-negative. We could force the counts to be non-negative by thresholding them to 0 or 1 and this would ensure that the released probabilities are defined. However, consider what happens to the counts when some features are sparse and privacy is stringent. When the added noise is big in comparison with the counts for a sparse feature, the signal in the data will be overwhelmed. Such feature will seem deceivingly informative and using it will degrade performance. We address both of these issues by proposing a thresholding heuristic, which we now place into context with the following concrete example.

Consider a scenario in which the count for a feature in one class is 200 and the count for that feature in the second class is 100. Let the dimensionality of this two-class problem be 100. An 0.1-differentially private mechanism could sample a noise vector $(-150, 900)$. The new counts will be $(50, 1000)$, making the likelihood of the second class 20 times bigger than the likelihood of the first class. To avoid this issue, we need to ensure that the noise is smaller in magnitude than the counts.

We propose that the perturbed counts are thresholded to a constant of the order of the noise. In particular, the noised counts are thresholded such that the threshold is bigger than the noise added most of the time. It is important to note that thresholding in this way may result in ignoring rare but informative features, because their counts may not be large enough even in the presence of a lot of data. Thus, several thresholds should be experimented with, but based on our experiments, the 65% percentile of the noise distribution Lap(0, $\frac{\delta}{\varepsilon}$) performs well:

$$thresh = -\frac{\Delta}{\varepsilon}\log(2(1-0.65)). \tag{3.15}$$

For example, when $\varepsilon = 1$ and $D = 600$ this is 214 and when $\varepsilon = 0.1$, it's 2140.

Releasing the class probabilities Releasing the class probabilities $\boldsymbol{\pi}$ is straightforward, since given the released $\boldsymbol{M}$, it is a post-processing step. In particular, the count for a class is given by the sum of the number of times each feature is off and on in this class. Because of the noise, this summation results in different counts for each feature. To obtain only one count per class, we take the mean of all sums. Taking the mean ensures that noise in the positive and negative direction cancels out on average and we get close to the true class count.

3.3.3 Privacy proof and analysis of the algorithm

Privacy guarantee

Theorem 3.3.1. Algorithm 1 satisfies ε-differential privacy.

Proof. Previously, we showed that the sensitivity of the Manhattan distance metric employed by this algorithm is D, the dimensionality of the data:

$$\Delta = D. \tag{3.16}$$

The algorithm releases $(\tilde{\boldsymbol{N}}^{(1)}, \tilde{\boldsymbol{N}}^{(0)})$ by sampling a Laplace distribution centered at the true values of the counts in each direction:

$$dP_X(\tilde{\boldsymbol{\theta}}) \propto \exp\left(-\frac{\varepsilon}{D}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|_1\right), \tag{3.17}$$

The computation of $(\tilde{\boldsymbol{N}}^{(1)}, \tilde{\boldsymbol{N}}^{(0)})$ is ε-differentially private, following from Theorem 2.2.1. In order to release $\tilde{\boldsymbol{\theta}}(c,:)$ no additional noise needs to be added, as per Theorem 2.3.1. This completes the proof. □

Scaling with the amount of data The uncertainty in the estimate of the model parameters resulting from the privacy restriction can be reduced with additional data. Indeed, since the sensitivity is equal to the dimensionality, the amount of additional data required by the algorithm to achieve good accuracy scales linearly with the dimensionality as well. The sensitivity of the algorithm, and thus the amount of noise, are constant with respect to the dataset size N. In low-dimensionality settings, it is therefore reasonable to expect that when N grows the noise levels will be small compared with the counts, achieving good performance. In high-dimensionality settings, it may not be possible to obtain the necessary amount of extra data, since when features are sparse counts can still be too small compared with the amount of noise.

The limiting behavior of Algorithm 1 when N goes to infinity is desirable - the parameters will not be perturbed. This is achieved by reporting $\theta_{jc} = (N_{on} + \alpha)/(N + \alpha + \beta)$, where α and β are the pseudo counts for the positive and negative class, respectively.

Limiting behaviour with epsilon As the privacy parameter is relaxed to infinity, Algorithm 1 will add noise with 0 variance, i.e. the noise disappears.

3.3.4 Empirical analysis of the algorithm

Experiments are performed using the UCI datasets summarized in Table 3.1, detailed further in Appendix A. If necessary, each dataset has been preprocessed to make all features binary.

Name	Train set size	Test set size	Number of features	Majority baseline
XWindowsDoc	900	900	600	50%
Mushrooms	5687	2437	112	52%
Adult	32561	16281	123	76%
Newsgroup	5687	2437	100	57%
Newsgroup2	4663	1998	100	81%

Table 3.1: Table showing the details of the preprocessed UCI datasets used for experiments. The majority baseline is the proportion of the bigger class in the dataset, i.e. the size of the bigger class over the size of the whole dataset.

Unless otherwise stated, the privacy parameter epsilon is 0.1. The randomized procedure is repeated 10 times and the average accuracies are reported with standard errors.

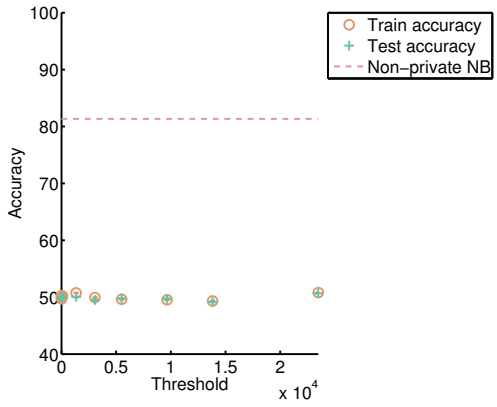
Sensitivity to the threshold As discussed earlier, thresholding was employed which ensures that parameters associated with features which are too rare will be ignored. It is important to see how this thresholding procedure affects the accuracy of the algorithm, since some of these rare features may be very informative prior to perturbation.

We conduct several experiments with the following thresholds: $0, 1, \log 2/\varepsilon, 10, 100$ followed by the 65%, 70%, 80%, 90%, 95%, and 99% percentiles.

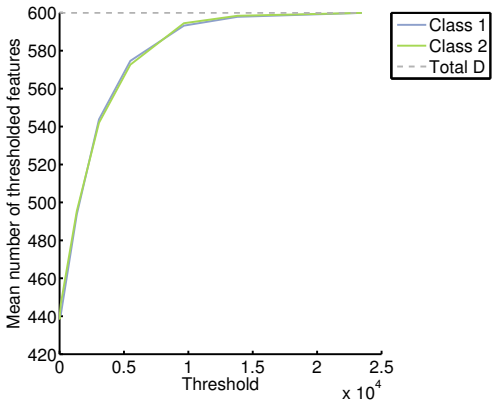
In figures 3.1a and 3.3a we show the accuracy of the algorithm on the XWindowsDoc and Newsgroup1 dataset, respectively, as a function of the threshold. No significant improvement in the accuracy is noticeable from using a different threshold. In Figure 3.2a we perform the same experiment on the Mushrooms dataset, where using a bigger threshold can improve the accuracy of the private algorithm by roughly 5%.

Figures 3.1b, 3.2b and 3.3b show the mean number of features for which the noisy count was thresholded. Thresholding a feature means it has the same value for all classes and thus becomes irrelevant to the classification decision. For the Mushrooms and Newsgroup1 datasets roughly 75% of the features were thresholded when the threshold is the 65th percentile of the noise distribution. In the XWondowsDoc dataset more features were thresholded, due to the larger number of sparse features in this dataset.

Figure 3.4 compares the mutual information, measured in bits, of features in the three datasets. The Mushrooms dataset has a few features which are very informative in contrast with the other two datasets, where the difference between the most and least informative features is not as drastic. For some values of the threshold, the algorithm

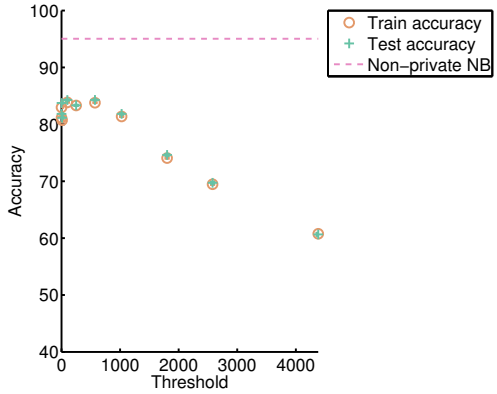


(a) Testing the sensitivity of Algorithm 1 to the choice of threshold t_{lower}. The accuracy does not change significantly. The fifth point corresponds to the 65% percentile.

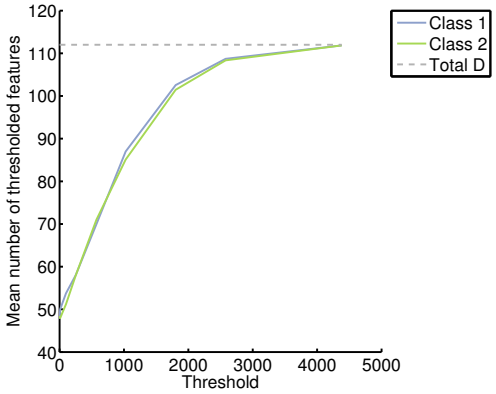


(b) The mean number of thresholded features as a function of the threshold.

Figure 3.1: Thresholding effect on the XWindowsDoc dataset The mean number of thresholded features corresponds to the number of features whose noisy counts were thresholded averaged over all runs of the private procedure. Figure generated by testThresholding.m.

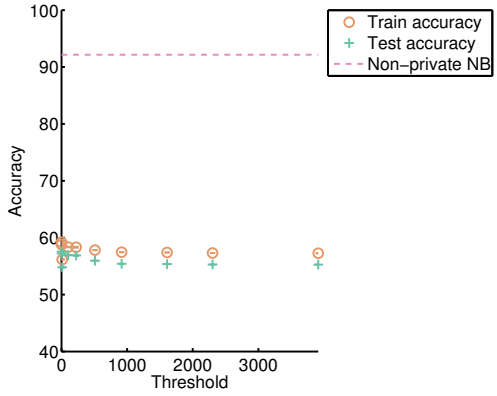


(a) Testing the sensitivity of the Algorithm 1 to the choice of threshold t_{lower}. Increase in accuracy results from choosing a smaller threshold. The fifth point corresponds to the 65% percentile.

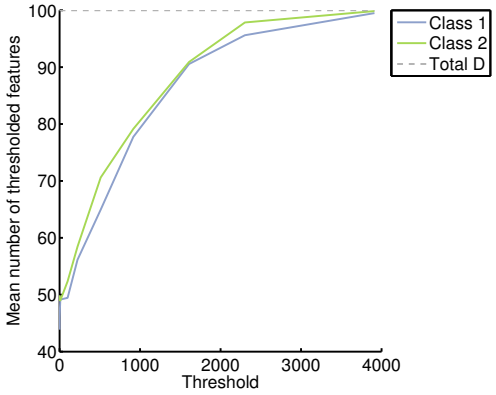


(b) The mean number of thresholded features as a function of the threshold.

Figure 3.2: Thresholding effect on the Mushrooms dataset The mean number of thresholded features corresponds to the number of features whose noisy counts were thresholded averaged over all runs of the private procedure. Figure generated by testThresholding.m.



(a) Testing the sensitivity of the Algorithm 1 to the choice of threshold t_{lower}. Virtually no effect is noticeable on the accuracy. The fifth point corresponds to the 65% percentile.



(b) The mean number of thresholded features as a function of the threshold.

Figure 3.3: Thresholding effect on the Newsgroup1 dataset The mean number of thresholded features corresponds to the number of features whose noisy counts were thresholded averaged over all runs of the private procedure. Figure generated by testThresholding.m.

manages to find the features that seem useful, while ignoring the ones for which the noise has trumped the signal. Figure 3.5 shows the mutual information, in bits, for features in the Mushrooms and Newsgroup1 dataset together with whether a feature was thresholded or not. We see that in the Mushrooms dataset a significant number of the informative features are preserved, whereas in the Newsgroup1 dataset a bigger proportion of the more informative features are thresholded and thus ignored. In the Mushrooms dataset, frequent features were more informative compared with the two other datasets. This allowed the thresholding procedure to ignore rare uninformative features, which were only contributing noise to the learned classifier. Thresholding could be useful in situations where there are a few rare uninformative features.

Different privacy settings This section investigates the privacy-accuracy trade-off of Algorithm 1.

Figure 3.6a shows the results of using the private classifier with different values of the privacy parameter ε on the balanced XWindowsDoc dataset. The total number of features for this dataset is 600 and the number of training samples is 900. An 0.1-differentially private Naive Bayes classifier needs to add noise with standard deviation of

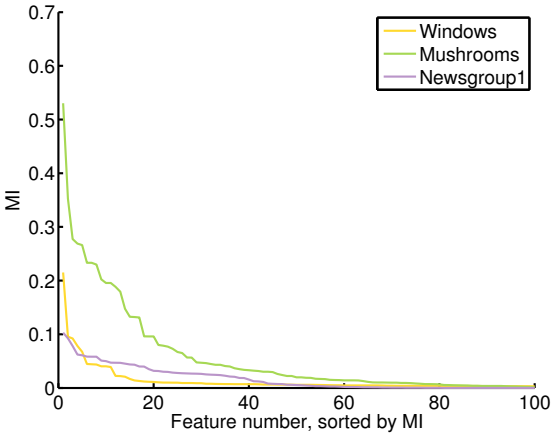


Figure 3.4: MI for the top 100 informative features The mutual information, measured in bits, for each feature in the XWindowsDoc, Mushrooms, and Newsgroup1 datasets, sorted, on the original data. Figure generated by testThresholding.m.

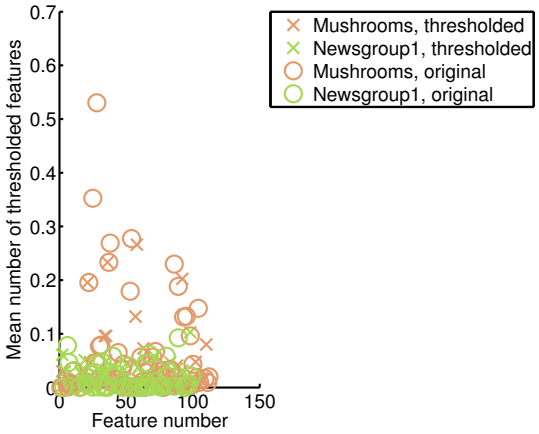
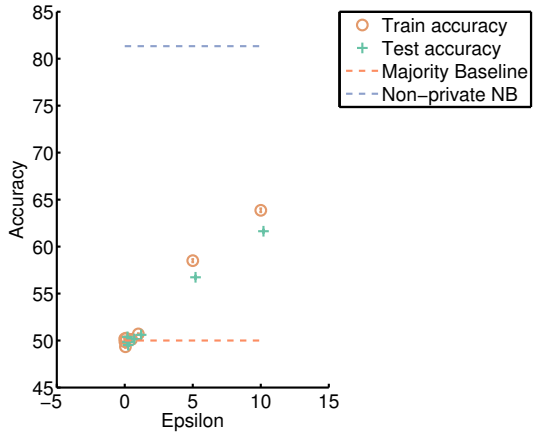
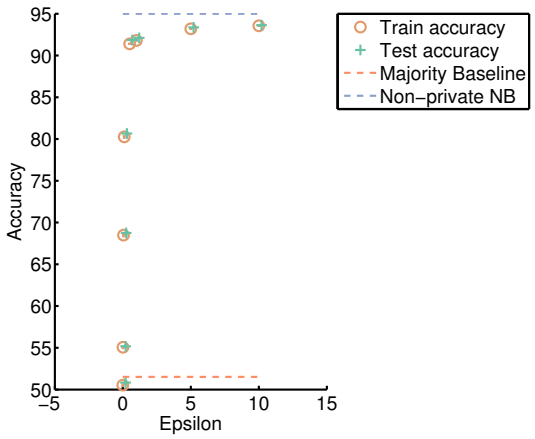


Figure 3.5: MI for each feature with thresholding information The mutual information, measure in bits, of the original features as a function of the feature number for one run of the algorithm. Crosses show that the feature was thresholded by $t_{lower} = -\frac{\Delta}{\varepsilon}\log(2(1-0.65))$. In the Mushrooms dataset the more informative features were also more frequent and thus not thresholded too often. Figure generated by testThresholding.m.



(a) XWindowsDoc The performance has suffered significantly due to the addition of noise and only beats the baseline for relatively big ε's. The experiment uses all 600 features and all 900 training samples. Figure generated by privateNBLaplaceNoiseVersion1.m.



(b) Mushrooms Reasonable performance could be obtained for privacy levels greater than 1. The experiment uses all 112 features and all 5687 training samples. Figure generated by privateNBLaplaceNoiseVersion1.m.

Figure 3.6: Scaling with ε Average accuracy for different epsilons, for 20 runs of the algorithm with a flat prior.

8500. Clearly, 900 samples are not enough to combat the stringent privacy requirements. The result is a classifier which makes random guesses. The noise cannot be compensated for and a lot more data is required for dimensionality as high as this.

Figure 3.6b shows the results of performing the same experiment on the Mushrooms dataset. Here, there is more data and less features, so even for relatively strict privacy requirements the algorithm can achieve good results. The increase in performance with the relaxation is much steeper than in the small N big D problem above. The private algorithm is indeed useful in scenarios where data is abundant, or the number of features is relatively small.

Implications of the Laplace Mechanism Further insight into the performance of private Naive Bayes can be gained from the sample complexity bound of the Laplace Mechanism. We apply Theorem 2.2.2 to the XWindowsDoc dataset, with $\varepsilon = 0.1$, 900 samples and 600 features. For a counting query with sensitivity 1, with probability 95%, no estimate will be off by more than an additive error of $\left(\log \frac{600}{0.05}\right)\left(\frac{1}{0.01}\right) \approx 94$. For datasets where features are sparse this is significant error capable of obscuring the signal in the data.

Scaling with the number of features As seen earlier, for small epsilons, the two classifiers trained were akin to doing random guessing. We now look at how the number of features affects the accuracy of Algorithm 1.

In the following experiments, features are ranked by their mutual information, measured in bits, with the class. Figure 3.7 shows the accuracy as a function of the number of features used, for the XWindowsDoc and Mushrooms datasets. On the horizontal axis is the mutual information of the least informative feature used: reading from right to left, the first result is the test accuracy of the classifier trained on the single most informative feature. The number of features used increases from right to left.

For non-private Naive Bayes (Figure 3.7a, XWindowsDoc dataset) selecting a few features can help increase accuracy by roughly 4%. For private Naive Bayes on the same dataset (Figure 3.7b) feature selection can result in accuracy close to the non-private, but using more than the first 7 or 8 features quickly degrades it quickly. In this figure, we see for the first time a private algorithm that performs above the baseline on this dataset. In Figure 3.7d, for the Mushrooms dataset, again the accuracy is improved by using the first few features. However, this time the decline in accuracy as a result of using more features is less steep, due to the bigger dataset size.

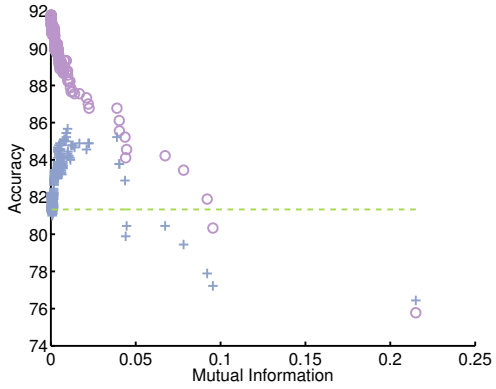
Privacy guarantee The above experiments show that using fewer features can drastically improve performance even on relatively high-dimensional datasets. However, the training procedure used is not ε-differentially private, because it violates Theorem 2.3.2. In particular, the feature selection step of sorting by mutual information does not preserve privacy, since the true mutual informations were used. A natural next step is private feature selection, which will be detailed in Section 3.5.

3.3.5 Relation to previous work

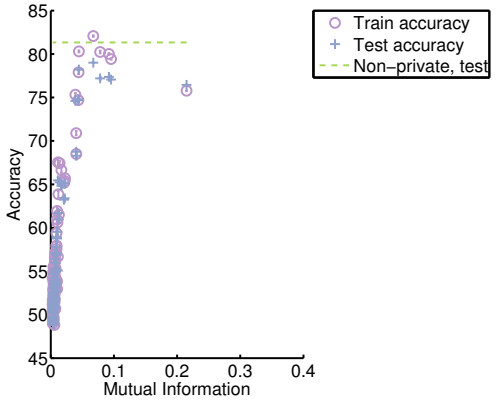
Differentially private Naive Bayes classifiers have previously been developed by Vaidya et al. (2013). Similarly to our work, their algorithm uses the Manhattan distance distance function and the perturbed version of the counts. Their fitting procedure, however, differs in two ways. First, they report the class probabilities as follows

$$nc'_j \leftarrow nc_j + \text{Lap}(0,1), \tag{3.18}$$

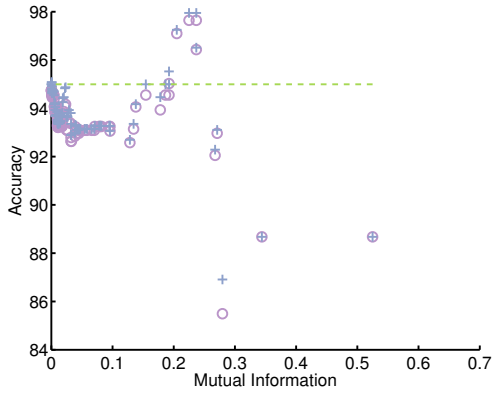
where nc_j is the number of examples in class c_j. With noise addition from $\text{Lap}(0,1)$, the ε used to report the probability of one class is 1; for C classes this subprocedure would be C-differentially private. Next, the number of times each feature appears in



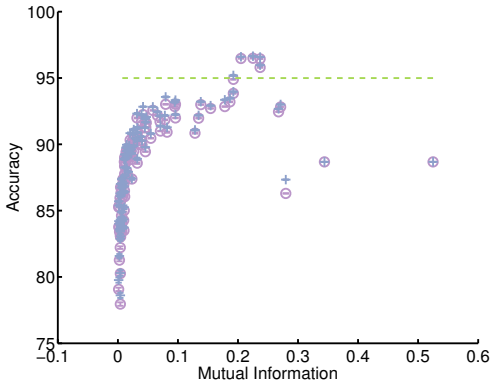
(a) Non-private version, XWindowDoc dataset Using only the first few most important features can increase accuracy by about 4%. Too many features, and the classifier overfits the data, as indicated by the small training error and big test error.



(b) Private version, XWindowsDoc dataset $\varepsilon = 0.1$; Using only the first few features increases the accuracy to 80%. When the number of features is increased, both the training and testing error rapidly increase.



(c) Non-private version, Mushrooms dataset Using the first most informative features achieves an increase of 3%.



(d) Private version, Mushrooms dataset $\varepsilon = 0.1$; Using only the first few features increases the accuracy to 97%. When the number of features is increased, both the training and testing error increase.

Figure 3.7: Accuracy as a function of the number of features used Horizontal axis shows the mutual information, measured in bits, of the least informative of the features used. The dashed line shows the test accuracy of the classifier trained on all features, circles and pluses show the train and test accuracy, respectively, for the specified number of features.

each class is reported as follows:

$$n_{kj} = n_{kj} + \text{Lap}(0, 1/\varepsilon). \tag{3.19}$$

Summing up the privacy budget according to the composition theorems 2.3.3 and 2.3.2, the total privacy budget used is $D\varepsilon$, where D is the number of dimensions. In total, the privacy guarantee of this algorithm is $D\varepsilon + C$, as opposed to the claimed ε. Instead, in our work, the scale of the noise distribution depends on the dimensionality of the dataset.

The second difference with our work is the way that negative counts and class probabilities are handled. In their implementation noise is resampled until the perturbed counts are non-negative. This is not a valid procedure, because the final distribution of the noise will be different from what was intended.

Finally, they test the implemented algorithm on a number of UCI datasets. Unfortunately, they only tested large values of the privacy parameter, $\varepsilon > 1$, whereas in practice a smaller value of ε is recommended. Their experiments fail to fully represent the real spectrum of useful private levels. Not surprisingly, for these relatively large values of the privacy parameter, their algorithm achieves good results on most datasets. Interestingly, on the Adult dataset their algorithm only achieves baseline performance even with values of ε bigger than 1. The Adult dataset seems particularly difficult for differentially private algorithms - even with the fairly relaxed privacy guarantee of their algorithm, the results for this dataset are discouraging.

3.3.5.1 Conclusion

Datasets where the number of samples is smaller than (or similar to) the dimensionality are common in the real world. As seen earlier, for high-dimensionality datasets, adding noise with scale proportional to the dimensionality can amount to adding fake data points on orders of magnitude greater than the available amount of data. So much fake data can obliterate the true model parameters.

Could the situation be improved? We could reduce the original dimensionality of the dataset, so that we have a lot more data than features. In this way we can hope to have more real data than fake data and hence more reliable perturbed parameters.

Before we go on to private feature selection, we discuss an alternative training procedure that could potentially improve the performance of the private Naive Bayes classifier and does not require feature selection.

3.4 Partitioning strategy

We propose a training strategy for private machine learning classifiers, where the original dataset is partitioned such that each partition is used to train one of the model parameters. For D model parameters, D separate private classifiers are obtained, each of which is trained on a subset of the data of size N/D. Intuitively speaking, the strategy exploits the fact that privacy noise is independent in each direction. Experiments with this strategy show that it can improve private Naive Bayes' performance in certain situations.

3.4.1 Experiments

We evaluate the partitioning strategy against the standard training methodology. We experiment with several datasets, including the XWindowsDoc, Mushrooms, and some synthetic datasets. To allow us to investigate the partitioning strategy with increasing amount of data, we have replaced the XWindowsDoc dataset with a synthetic proxy dataset. In particular, we generate 3600 train and 3600 test samples by sampling from a non-private Naive Bayes model trained on the original dataset. Similarly, we generate two synthetic datasets by sampling from the following two Naive Bayes models:

$$\boldsymbol{\theta}_{\text{dense synthetic dataset}} = \begin{bmatrix} \dots & \approx 0.8 & \dots \\ \dots & \approx 0.5 & \dots \end{bmatrix},$$

and

$$\boldsymbol{\theta}_{\text{sparse synthetic dataset}} = \begin{bmatrix} \dots & \approx 0.01 & \dots \\ \dots & \approx 0.03 & \dots \end{bmatrix},$$

where we have 2 classes and 300 features. The train set and the test set are both of size 3000, 30000, or 60000.

The setup for this experiment is as follows. The features for each dataset are sorted by their mutual information with the class. Then a certain number of features, shown on the horizontal line, is chosen. For each number of features, T, four classifiers are compared. First, in yellow, is the private classifier obtained by splitting the original dataset into T smaller datasets, training T separate one-feature models, and then combining them into one T-dimensional vector of model parameters. In light green circles, is the private classifier trained with all data with all features at the same time. In blue circles is its non-private counterpart. Finally, in orange pluses, is the non-private classifier trained by the partitioning strategy. Each marker shows the average accuracy over 50 runs of

the randomized procedure together with the standard error. The dimensionality of the large problem is gradually increased. The training set is constant size, so with more features the number of samples per feature in the partitioning strategy gets smaller.

Results are shown in Figure 3.8. For the XWindowsDoc, Mushrooms and the dense synthetic datasets, figures 3.8a, 3.8b and 3.8c, respectively, there is a noticeable improvement in the accuracy of the private classifier resulting from partitioning. The improvement is more noticeable for certain values of the dimensionality, because the loss in generalization performance for that number of samples is less damaging than the loss due to privacy.

For sparse datasets, figures 3.8d and 3.8e, using the partitioning strategy has a negative effect on performance, for both the non-private and private settings. This is due to the sparsity of the data: splitting results in even sparser features and more likely to be incorrectly estimated.

Finally, Figure 3.8f presents a setting where there is a slight positive effect of partitioning on a sparse dataset. In this setting, there is a lot of data per feature, but the privacy restrictions are more severe.

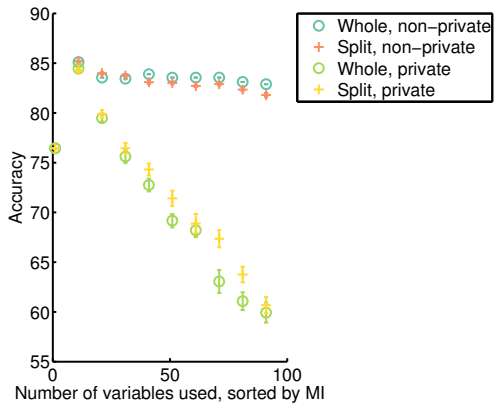
In conclusion, the partitioning strategy can slightly benefit the performance of a private classifier, as long as enough data is available to train for each feature separately. For dense datasets less data is usually needed and partitioning may prove beneficial. For sparse problems partitioning is less useful.

3.5 Differentially private feature selection

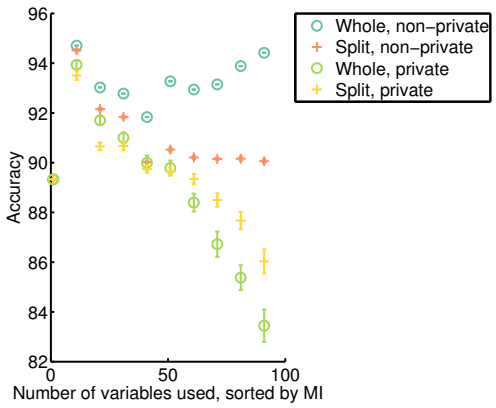
As shown previously, dimensionality reduction could be a viable option for improving the performance of differentially private Naive Bayes on higher-dimensional datasets. The problem which the above considerations have left unsolved, however, is differentially private feature selection.

3.5.1 Input and output

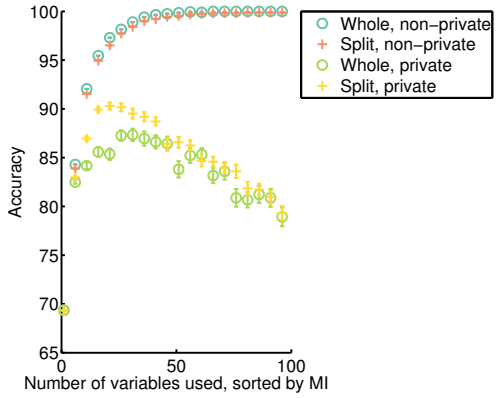
A private feature selection algorithm takes as input the data allocated for feature selection, a privacy parameter ε, specifying how much of the privacy is allocated for feature selection, and the number of features to be selected, T. The algorithm outputs a vector which specifies which features are selected. This is a vector of length T of integers in the range $[1, D]$. Here, we will prefer the one-hot-encoding, where the output



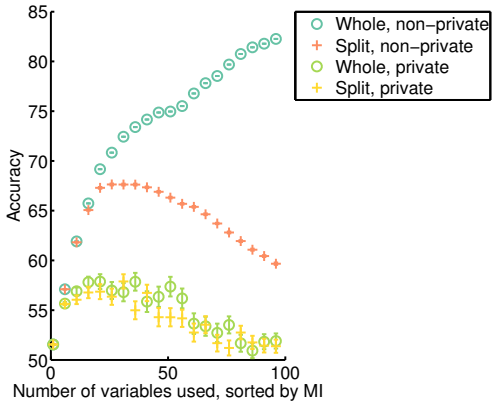
(a) XWindowsDoc dataset proxy No privacy: partitioning makes no difference; privacy: splitting the dataset could be useful.



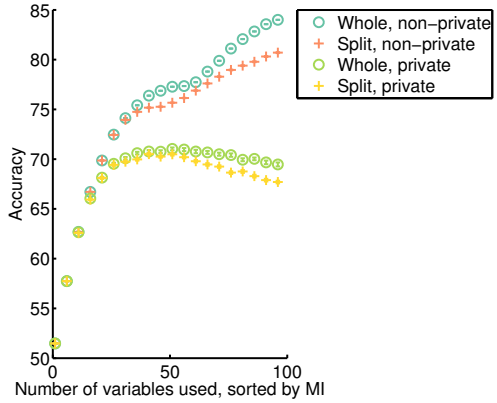
(b) Mushrooms dataset No privacy: decrease in accuracy; privacy: partitioning could be useful.



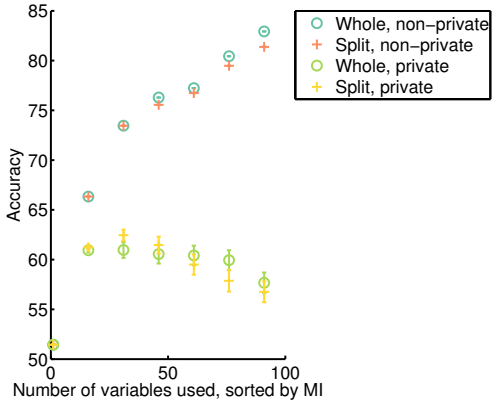
(c) Dense synthetic dataset of size 1000 and dimension 500 Noticeable improvement through partitioning.



(d) Sparse synthetic dataset of size 3000 No improvement from partitioning.



(e) Sparse synthetic dataset of size 30000 No improvement from partitioning.



(f) Sparse synthetic dataset of size 60000, $\varepsilon = 0.01$. Questionable advantage from partitioning.

Figure 3.8: Partitioning strategy In yellow, is the private classifier obtained by splitting the original dataset, then training several one-feature models and combining them. In light green circles, is the private classifier trained normally. In blue circles is its non-private counterpart. Finally, in orange pluses, is the non-private classifier trained by the partitioning strategy. Partitioning is more useful for dense datasets.

is $\{0,1\}^D$, i.e. a binary vector of length D.

There are two alternative ways in which the data and privacy budget can be used when feature selection is performed prior to training. The first input option for feature selection is to use all of the training data available, but dedicate only a portion of the total privacy budget; the rest of the privacy budget and all of the training data is then passed to the training procedure. The second option is to use a subset of the training data and ε equal to the total privacy budget; the rest of the data and the total ε is then passed to the training procedure.

Choosing the strategy to employ is problem dependent. It is suggested that domain knowledge and information about the size of the available training set is used to make this decision. In the presentation of feature selection algorithms it will be assumed that this decision is made in advance. In the experiments, it will be specified what ε and data are used as needed.

3.5.2 Noisy variable ranking

The most basic approach to feature selection is variable ranking. It is often used as auxiliary selection mechanism due to its scalability, simplicity, and satisfactory empirical performance (Guyon and Elisseeff, 2003). Referred to as a filter method, variable ranking is independent of the choice of classifier.

The general procedure for variable ranking is as follows. Using a certain scoring function, computed from the values x_{nd} of feature $\mathbf{x_d}$ and y_n, $n = 1,\ldots,N$. A high score means that $\mathbf{x_d}$ is a useful variable for prediction. Sorting the variables in decreasing order of their score, the top T variables are selected for use in training a classifier.

Other common criteria for variable ranking are correlation, individual predictive power (the performance of a classifier built using a single variable), and mutual information. Due to its scalability, noisy variable ranking is a promising first step towards effective private feature selection.

Distance metric As usual in the design of differentially private algorithms, we have to select the score function and the distance metric on its output space. An important decision also regards the stage of the algorithm at which the noise is added.

The score function quantifies the "goodness" of the output - in this case the usefulness of each feature. Since the mutual information is a common feature selection criterion in the non-private setting, it is worth investigating as the quality score function of a private feature selection algorithm. The distance metric on the output space will be

the Manhattan distance.

3.5.2.1 Selecting features by fitting another model

A natural first attempt at noisy variable ranking is as shown in Algorithm 2. The algorithm fits a private Naive Bayes model, then computes the mutual information of each feature using the perturbed counts; then finally sorts the mutual informations and reports the top T. Algorithm 2 adds noise in the beginning, as in input perturbation.

Algorithm 2 Noisy variable ranking by fitting another model

1: **procedure** SELECTFEATURES(X, y, ε, T)
2: $\tilde{\boldsymbol{\theta}} = fitPrivateNB(X, y, \varepsilon)$
3: Compute $\tilde{I}(\mathcal{X}, y)$, a vector of length D
4: $sorted = sort(\tilde{I}(\mathcal{X}, y))$
return One-hot encoding of $sorted(1:T)$

The algorithm is $\varepsilon-$differentially private, since $fitPrivateNB$ is ε-differentially private and steps 3 and 4 are simply post-processing.

As already shown, fitting a private Naive Bayes model to high-dimensional data is not straightforward unless abundant data is available. Moreover, only part of the total privacy budget (or a subset of the training data) is available for selection. Thus, this feature selection procedure is unlikely to be effective and a more privacy efficient solution is needed for high-dimensional problems.

3.5.2.2 Releasing the mutual information

Instead of input perturbation, we now consider output perturbation - releasing perturbed mutual informations, which have been computed using true counts. The algorithm is shown in 3.

Algorithm 3 Noisy Variable Ranking by releasing I

1: **procedure** SELECTFEATURES(X, y, ε, T)
2: Compute $I(\mathcal{X}, y)$, a vector of length D
3: $\Delta = D\left(\frac{1}{N}\log N + \frac{N-1}{N}\log\frac{N}{N-1}\right)$
4: $\tilde{I}(\mathcal{X}, y) = I(\mathcal{X}, y) + Laplace(0, \Delta/\varepsilon)$
5: $sorted = sort(\tilde{I}(\mathcal{X}, y))$
return One-hot encoding of $sorted(1:T)$

$X_j \backslash Y$	0	1
0	1	0
1	0	0

(a) An example of the minimum mutual information.

$X_j \backslash Y$	0	1
0	$\frac{N-1}{N}$	0
1	0	$\frac{1}{N}$

(b) Adding one new sample and its effect on the mutual information.

Table 3.2: Sensitivity of the Mutual Information

Sensitivity As usual, it is necessary to compute the sensitivity of the score function (Zhang et al., 2014). This is done on per feature basis. Since the mutual information is always non-negative and binary variables are assumed, the minimum possible mutual information is 0, for example as for distribution 3.2a. The mutual information is minimal since there are only examples in one of the classes - i.e. the entropy of the class label is 0.

In order to obtain the maximum change in the mutual information of the X_j and Y as a result of one additional sample, it has to be added to the opposite class. Moreover, the new example has to have $X_j = 1$ as opposed to 0, because when X_j completely determines the value of Y, the mutual information is maximum.

Alternatively, the opposite example could be constructed, where starting with the maximal mutual information and adding one example. The result will be the same. Finally, the maximum change in the mutual information, and thus the sensitivity is (Zhang et al., 2014):

$$\Delta_I = \frac{1}{N}\log N + \frac{N-1}{N}\log\frac{N}{N-1}. \tag{3.20}$$

Is this sensitivity good? How does output perturbation compare with input perturbation when the goal is to release the mutual informations?

A plot of the sensitivity as a function of the dataset size is shown in Figure 3.9. In the limit of infinite data the sensitivity is 0, which is a desirable property. When designing a private mechanism and using the Manhattan distance, we are concerned with the scale of the noise and the ratio

$$\frac{\varepsilon}{\Delta_I}\|I(X_j,Y) - \tilde{I}(X_j,Y)\|_1. \tag{3.21}$$

The mutual information of binary variables is in the interval $I(X_j,Y) \in [0,1]$. Since ε is fixed to a small constant, we would like to compensate for it, by making $\|I(X_j,Y) - \tilde{I}(X_j,Y)\|_1/\Delta$ big. The faster Δ/ε approaches 0, the better. If there are around 100 samples and $\varepsilon = 0.1$, the variance of the noise distribution will be roughly 0.5. This

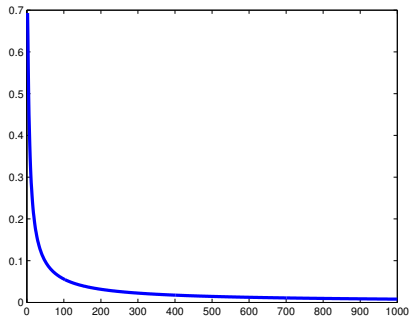


Figure 3.9: Sensitivity of mutual information as a function of the number of examples N.

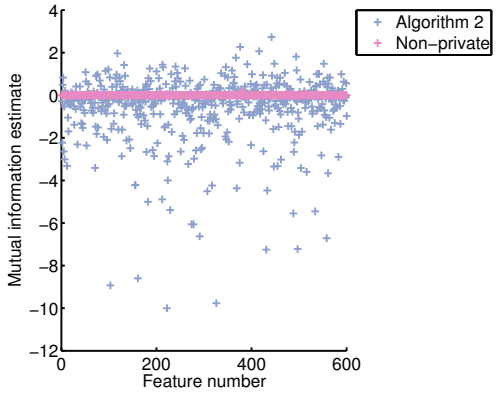
might seem small given the range of the mutual information, but it is often the case in practice that mutual information is fairly small. Thus, using the mutual information as a score function is unlikely to produce good results. This remains to be shown empirically.

Privacy guarantee of Algorithm 3 Noise is added for each feature independently with scale $\Delta_I/(\varepsilon/D)$. The informativeness of each feature is released in an ε/D-differentially private manner as a direct consequence of using the Laplace mechanism. From the summation property of differential privacy it follows that releasing the informativeness scores of all D features is ε-differentially private.

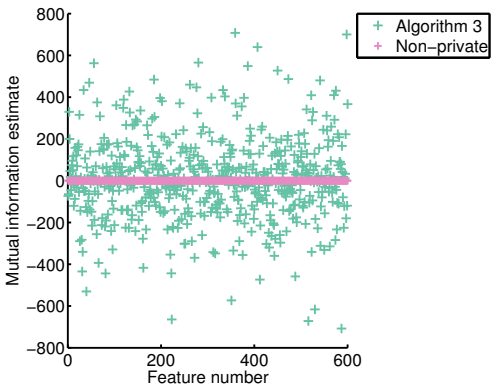
Composing this algorithm with any $\varepsilon-$differentially private fitting procedure will be 2ε-differentially private if the same training data is used; or $\varepsilon-$differentially private if the data was partitioned for feature selection and training.

Comparison of algorithms 2 and 3 Figure 3.10 compares the amount of noise in the estimate of the mutual information for both algorithms, applied on the XWindows-Doc dataset ($D = 600, N = 900$), taking half of the available privacy budget. Clearly, both algorithms are ineffective – the scale of the noise is orders of magnitude greater than the scale of the mutual information. Algorithm 3 is especially wasteful with privacy noise variance in the hundredths. For a more formal treatment of this issue, Theorem 2.2.2 could be applied to derive the probability of certain errors. Both Algorithms 3 and 2 release a distribution over all D features. The privacy budget needs to be divided by D, making both algorithms hopeless in large D settings.

Furthermore, the principle of minimum information suggests that unnecessary detail is released: the mutual information is a real number, giving a very precise answer to how informative a feature is. What is actually needed for feature selection is a binary mask vector that specifies which feature is selected. Intuitively, it should be enough



(a) Algorithm 2 The privacy noise is too much to handle.



(b) Algorithm 3 The noise variance is enormous.

Figure 3.10: Perturbed mutual information, $\tilde{I}$, XWindowsDoc dataset The true versus the perturbed mutual informations, in bits, for all 600 features, for $\varepsilon = 0.05$. Both algorithms are too wasteful.

to just release a binary decision, without explaining why this decision was reached. Exploiting this intuition results in feature selection algorithms that are more privacy efficient.

3.5.3 Iterative feature selection

This section investigates an iterative strategy for private feature selection which selects T out of D original features. Although strictly speaking not ε-differentially private, the algorithm is considered with the purpose of showing that a naive iterative procedure for selecting T features has unsatisfactory performance, even with relaxed privacy restrictions. We take the mutual information heuristic, however the argument generalizes to other metrics.

Outline The algorithm begins by dividing the total privacy budget by T, the number of features to be selected. The true mutual informations for all features are computed; then sorted and shifted. The main loop of the algorithm considers features in decreasing order of their mutual information with the class. For each feature, the probability of selecting it is computed by using the exponential mechanism with quality score the mutual information. A binary decision is then made using the calculated probability of success. The loop continues until T features are selected.

Privacy guarantee A feature is selected by using the exponential mechanism with $\varepsilon' = \varepsilon/T$; therefore this procedure is ε'-differentially private. If all of the first T features

Algorithm 4 Iterative Sampling with MI

procedure SELECTFEATURES(X, y, ε, T)
  $\varepsilon' = \varepsilon/T$
  Compute $I(\mathcal{X}, y)$, a vector of length D
  $sortDescending(I)$
  $I = I - I_{T+1}$
  $t = 0$
  $selected = zeros(D, 1)$
  while $t < T$ **do**
    $I_t = top(I)$
    $p_t = \frac{1}{1+\exp(-\frac{\varepsilon'}{2\Delta} I_t)}$
    $X_j \sim Ber(X_j | p_t)$
    if $X_j == 1$ **then**
      $t = t + 1$
      $selected = selected \cup$ one-hot encoding of X_j
  return $selected$

are selected (and the loop makes exactly T iterations), the total privacy guarantee of the algorithm would indeed by ε.

However, not all features considered are guaranteed to be selected. Therefore, the loop will execute more than T times, breaking ε-differential privacy. In order to be able to guarantee ε-privacy we can take $\varepsilon' = \varepsilon/D$, i.e. using the upper bound on the iterations. However, as already seen, for large D this strategy is hopeless.

The true privacy guarantee of the algorithm can be computed by taking the actual number of loop iterations and multiplying that by ε/T. It is also possible to modify this procedure, such that different values of ε' are used for different features.

Shift The mutual information for binary features is in the range $[0, 1]$. However, for real datasets, the values of the mutual information can be significantly closer to 0 than 1, for example, for the XWindowsDoc dataset, the top 5 most informative features are in the range $[0.07, 0.2]$. As seen previously, the exponential mechanism reduces to the sigmoid$(\frac{\varepsilon'}{2\Delta} I)$ when the output is binary. Rescaling would allows us to avoid probabilities that are too close to each other. Unfortunately, the sensitivity has to be rescaled too, so multiplying by a constant will make no difference. Instead, the mutual informations are only shifted to the left by I_{T+1}. This allows the most T informative features to be on the right side of 0, and more likely to be selected.

Efficiency The running time of Algorithm 4 is $O(D)$, since the loop can execute a maximum of D times, ignoring the complexity of the sort subroutine.

Will Algorithm 4 be effective?

Flatness of distribution over features Consider Figure 3.11. In the non-private context, the mutual information can provide a useful distribution over features. In the private context, however, scaling the mutual information achieves nothing since any constant is cancelled by the sensitivity. The distribution over features will be too flat. What is necessary is a quality score function which is naturally large in value compared to its sensitivity; first to be able to compensate for the small value of the privacy parameter; second to allow informative and non-informative features to be well-separated. Such feature will be introduced in Section 3.5.4.

Privacy budget efficiency Another issue with Algorithm 4 is its low privacy budget efficiency. Although significantly better than previous naive versions, it is still possible to improve upon the privacy budget used per feature. Due to the iterative nature of the algorithm, we had to split the total budget for feature selection amongst the T selected features. What if we could obtain the bit mask vector by sampling only once, using the total privacy budget? We propose a new dynamic programming procedure for sampling in this manner in Section 3.5.5.

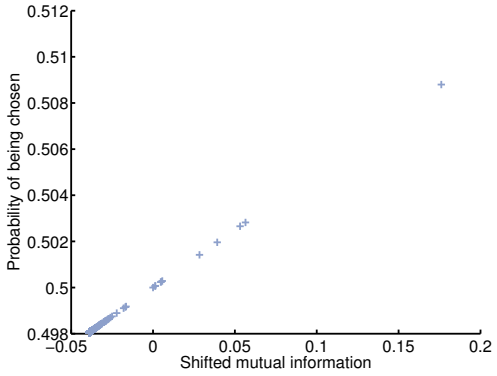
3.5.4 Count-based distance metric

This section motivates and presents a new heuristic for private feature selection which addresses the shortcomings of methods based on the mutual information.

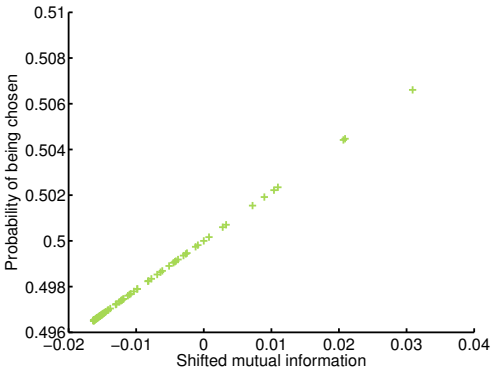
The range of the mutual information compared with its sensitivity is not big enough to compensate for the privacy restrictions. As a result, the distribution that the exponential mechanism defines over the features is too flat. In contrast, counting queries have been successfully used with private mechanisms. This is because the sensitivity of counting queries is usually 1, but their range is much bigger, depending on the problem setting. We exploit counting queries for private feature selection with good success. We focus on binary data, but extensions are easily imaginable.

3.5.4.1 Definition

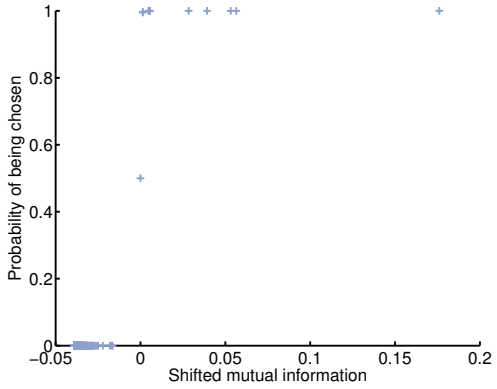
A feature is informative if it behaves differently for the different classes, for example it appears with different probability in the different classes. For binary features in a balanced two-class problem this could mean the feature is 1 more often in one of the



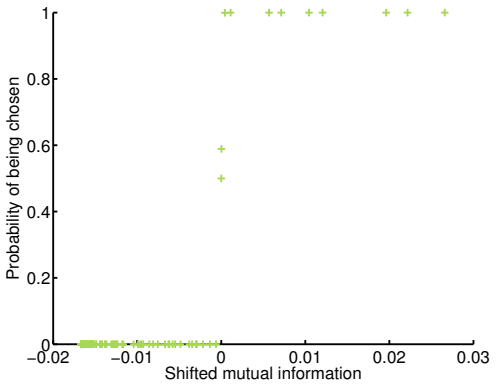
(a) XWindowsDoc dataset, with privacy.



(b) Newsgroups2 dataset, with privacy.



(c) XWindowsDoc dataset, no privacy.



(d) Newsgroups2 dataset, no privacy.

Figure 3.11: Mutual information score The probability of a feature being chosen as a function of its mutual information. Under the differential privacy restriction (top row) the probabilities of all features are close to 0.5, so the resulting Categorical distribution over features will be too flat. In the non-private setting (bottom row) the distribution over features is much more peaked and there is a clear separation between the good and the bad features. The number of features to be selected is 10 and the privacy level is $\varepsilon = 0.005$ per feature. The mutual information is measured in bits.

classes. When the classes are imbalanced it is not enough to compare the on-counts, since they could be the same. However, the proportions of on-counts in the two classes could be very different, simply because one class is bigger. The proposed heuristic considers both the number of times a feature is on in a class and the number of times it is off.

The usefulness of a feature, given a dataset, is determined as follows:

$$q(X_j) = \max\left(\left\|\theta_{j1}^{(1)} - \theta_{j0}^{(1)}\right\|_1, \left\|\theta_{j1}^{(0)} - \theta_{j0}^{(0)}\right\|_1\right), \tag{3.22}$$

where the superscript means on or off and the subscripts give the feature number and class. The heuristic computes the difference between the number of times the feature is on in the first class and the number of times the feature is on in the second class. The difference for the off-counts is also computed and then the maximum of those is taken. In other words, the differences per count type are compared. When the classes are balanced the two differences will be the same, since the off-counts are the total size of the class minus the on-counts. When imbalanced, the heuristic would select the bigger difference, giving more weight to a feature that is positively or negatively correlated with the class. A procedure using the proposed distance measure will prefer features for which at least one of the on-counts or off-counts are significantly bigger in one of the classes.

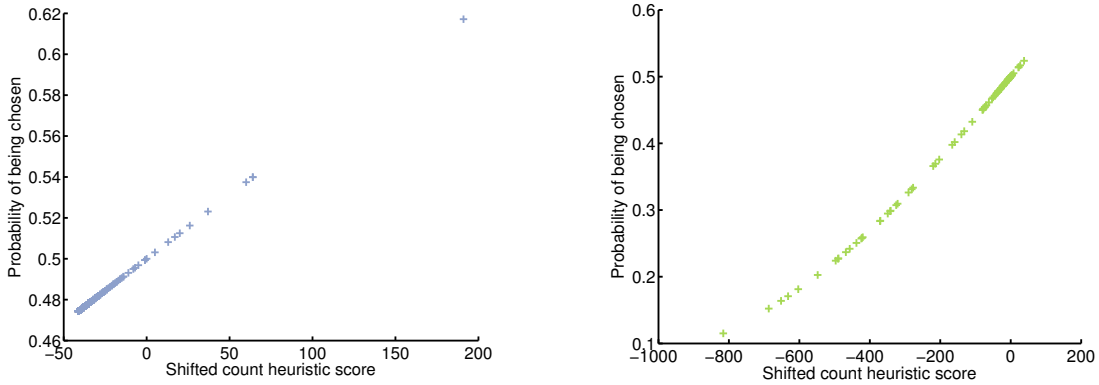
Further intuition into why this is a good measure could be gained from considering other distances. For example, instead of taking the differences per count type, we could compare differences per class, i.e. for each class subtract the number of times the feature is on and off in that class. Now, a most informative feature (one which has the feature only turned on in the positive class) and a least informative feature (one which is always on) will have the same scores. Taking the differences per count type avoids this issue.

3.5.4.2 Sensitivity

In the most informative case, where the feature is always on in class 1 and always off in class 0, the distance will be the size of the bigger class.

$$q(X_j) = \max(\|\theta_{j1}^{(1)} - 0\|_1, \|0 - \theta_{j0}^{(0)}\|_1) = \theta_{j1}^{(1)} : \tag{3.23}$$

If a negative sample is added to the negative (smaller) class, the maximum will not change . If the two classes are the same size, the maximum will change by one. If a positive example is added to the bigger class, the max will again change by one. If we



(a) XWindowsDoc dataset.

(b) Newsgroups2 dataset.

Figure 3.12: Counts heuristic score Plots show the probability of a feature being chosen as a function of its counts heuristic score. The probabilities are now further apart compared with using the mutual information, so the resulting Categorical distribution will be more peaked. The number of features to be selected is 10 and $\varepsilon = 0.005$ for each feature.

add negative example to the positive class, the max will likely not change, but if it does, it will change by one. Similarly, in the least informative case, the distance will be the difference between the size of the bigger and the size of the smaller class:

$$d(z_1, X) = \max(\|\theta_{j1}^{(1)} - \theta_{j0}^{(1)}\|_1, \|0 - 0\|_1). \tag{3.24}$$

Whatever example we add, this can change by at most one. Not surprisingly, the sensitivity of the feature quality score function is

$$\Delta = 1. \tag{3.25}$$

3.5.4.3 Distribution

Figure 3.12 shows the probability of a feature being selected as a function of its counts heuristic score. The probabilities are computed by Algorithm 4 but using the counts heuristic instead of the mutual information. Now, the probabilities are not as close to each other, which means the resulting distribution will be more peaked. The improvement on the Newsgroup2 dataset is more significant.

3.5.4.4 Relation to other work

Stoddard et al. (2014) recently independently suggested using what they refer to as Purity Index for feature selection. The purity index heuristic they offer is exactly the

same as the count-based heuristic featured in this work. In addition, they also evaluate the mutual information as a feature selection heuristic and state that it may result in too noisy estimates, which coincides with our findings.

The authors further provide three algorithms for private feature selection. The first algorithm is similar to the noisy variable ranking we have described: it perturbs the feature scores using the Laplace mechanism and then picks the features which rise above a certain threshold. To improve upon the results of this algorithm, they suggest a second approach, where the features are clustered according to their scores using differentially private k-means clustering. After private clusters are computed, a representative score is computed for each cluster, and features are selected only from the best clusters. Finally, they present a third algorithm, called private threshold testing, which perturbs a threshold rather than the scores of all features. Although an interesting idea, it is difficult to see how this would be applied in practice, since there is no explanation of how the threshold could be chosen.

We now detail an algorithm for private feature selection, which enjoys better privacy guarantees than the algorithms presented thus far.

3.5.5 Privacy efficient sampling

This section proposes a dynamic programming algorithm for feature selection, improving upon the privacy budget efficiency of the selection algorithms discussed previously.

Previous methods have had to divide the privacy budget available for feature selection amongst the total number of features D. We have also made an attempt at an iterative algorithm for feature selection, but shown that the task of selecting exactly T features with privacy degrading according to T rather than D is non-trivial. We now show a dynamic programming solution that selects all T features at once with good privacy guarantees.

The algorithm samples a bit mask vector $\boldsymbol{m}$ of length D from the following mask distribution:

$$p(\boldsymbol{m}) \propto \mathbb{1}\left(\sum_{d=1}^{D} m_d = T\right) \exp\left(\frac{\varepsilon'}{2\Delta} \sum_{d=1}^{D} m_d q_d\right), \tag{3.26}$$

where T is the number of features to be selected, D is the original dimensionality of the data, $q_d = q(X_d)$ is the quality score for feature X_d, and $\mathbb{1}()$ is the indicator function. The normalization constant of the above is

$$Z = \sum_{m} \prod_{d=1}^{D} \exp\left(\frac{\varepsilon'}{2\Delta} m_d q_d\right), \tag{3.27}$$

where the sum is over all masks with T ones.

As typical in dynamic programming algorithms for sampling, Algorithm 5 makes a forward and a backward pass. (For some function calls some arguments are omitted for brevity, but it should be obvious what these are.) The forward pass computes the normalization constant Z in 3.27 by breaking it down into smaller sums over sub-masks with fewer than D terms. In particular, the function computeSum(t,d) computes the sum over all sub-masks with $d \leq D$ elements and $t \leq T$ ones:

$$s_{(t,d)} = \sum_{\boldsymbol{m}_{1:d}} \prod_{i=1}^{d} \exp(m_i q_i), \text{where } \sum_{i=1}^{d} m_i = t. \tag{3.28}$$

This sum is of course, built up of smaller sums, so the function is recursive. The intermediate sums can be reused and are memoized in *hmap*, a hash table mapping (t,d) combinations to $s_{(t,d)}$s. The base cases are as follows. If the number of features still to be selected is 0, then the sum is 0, since empty masks are not accepted. If the number of features still to be selected is the same as the number of features to be considered, then the sum contains one term only - the mask with all 1s. The sum $s_{(t,d)}$ is composed of the sum where the dth bit is 1 and the sum where $\boldsymbol{m}(d) = 0$. Thus, $Z = \text{computeSum}(T,D)$. Numerical issues due to the magnitude of the counts are avoided by tracking the log sums.

A bit mask is obtained by backtrack sampling. The algorithm begins by sampling the final D bit of the mask. The probability that the Dth bit is on is equal to the proportion of the total sum Z which is made up by masks with $\boldsymbol{m}(D) = 1$. $\boldsymbol{m}(d)$ is then sampled from a Bernoulli with the computed probability.

Depending on the Dth bit, there are two options: there are still T bits left to turn on, or $T-1$. The probability that the dth bit is on with t bits to still turn on is the proportion of the sum $s_{(t,d)}$ where that bit is on. The algorithm is fully fledged out in 5.

Privacy guarantee Algorithm 5 defines a distribution over the possible mask vectors using a mask quality score, which specifies the "goodness" of feature masks. The privacy guarantee of the algorithm follows trivially from the use of the exponential mechanism, but the sensitivity of the mask quality score needs to be derived.

The quality score for a feature map of length D is defined as

$$\boldsymbol{q}_{\boldsymbol{m}} = \sum_{d=1}^{D} m_d q_d, \tag{3.29}$$

where $q_d = q(X_d)$ is the quality score for feature X_d, m_d is the corresponding bit in the feature mask and $\sum_{d=1}^{D} m_d = T$, i.e. exactly T of the D bits in the mask are on.

The sensitivity of the mask quality score is defined as the maximum change in the sum above as a result of adding a new sample to the dataset. When a new point is added, each of the q_d's can change by at most Δ_{q_d}. The mask vector does not change, because it is the output of the exponential mechanism. Finally, only T of the mask bits are on, and therefore,

$$\Delta_{\boldsymbol{q_m}} = \max_{X \sim X'} \left(\left\| \sum_{d=1}^{D} m_d q_d - \sum_{d=1}^{D} m_d q'_d \right\|_1 \right) \leq T \Delta_{q_d} \tag{3.30}$$

where $q_d = q(X_d)$, $q'_d = q(X'_d)$, and Δ_{q_d} is the sensitivity of q_d, i.e. the maximum change in each of the q_ds as a result of adding one example.

For the count heuristic the sensitivity of the quality score per feature is 1 and therefore the total sensitivity of the feature map quality is T. For the mutual information heuristic, the sensitivity is

$$T \left(\frac{1}{N} \log N + \frac{N-1}{N} \log \frac{N}{N-1} \right).$$

3.5.6 Related work

Sparse vector technique The composition theorem of differential privacy (see Theorem 2.3.2) state that a mechanism used for answering an adaptable list of queries should have a privacy parameter that degrades proportionally to the number of answered queries. The sparse vector technique (Dwork and Roth, 2014) was designed for online query-answering where the list of queries is long and the sensitivity of each query is one. It embodies the intuition that in some situations we are only interested in whether a response to a query lies above or below some threshold, and when this is the case, the privacy parameter should degrade with the number of positively answered queries. The technique is called sparse because it is most useful when the number of positive answers is much smaller than the total number of possible queries.

We can think of univariate feature selection for high-dimensional datasets as answering a long list of adaptable queries, where there is one query per feature and the response is whether or not the feature is selected.

The algorithm is build up of calls to a sub-routine called AboveThreshold. Above threshold takes a threshold fixed in advance, an adaptable list of sensitivity-one-queries, the data, and the privacy parameter ε', and outputs binary answers of whether the threshold was exceeded. It adds noise to the true response and compares the result to the threshold. For technical reasons, the algorithm works with a noise version of the

Algorithm 5 Privacy efficient sampling

procedure SAMPLEMASK($\varepsilon, \Delta, q, T, D$)
  $hmap = \{\}$, an empty hashmap
  $\boldsymbol{m} = zeros(D, 1)$
  $t = T$
  for $d = D : 1$ **do**
    if $t <= 0$ **then**
      $p = 0$
    else if $t > d$ **then**
      $p = 0$
    else
      $p = \frac{\exp(\text{computeSum}(t-1,d-1)+\frac{\varepsilon'}{2\Delta}q_d}{\exp(\text{computeSum}(t,d))}$
    $\boldsymbol{m}(d) = Bernoulli(p)$
    $t = t - mask(d)$
  return $\boldsymbol{m}$

procedure COMPUTESUM($t, d, \varepsilon, \Delta, q, hmap$)
  $key = \text{makeKey}(t, d)$
  if $hmap$.containsKey(key) **then**
    $s = hmap$.get(key)
  else if $t == 0$ **then**
    $s = 0$
  else if $t == d$ **then**
    $s = \frac{\varepsilon'}{2\Delta}\sum_{j=1}^{d} q_j$
  else
    $s_t = \text{computeSum}(t, d-1)$
    $s_{t-1} = \text{computeSum}(t-1, d-1) + \frac{\varepsilon}{2\Delta}$
    $s = \log(1 + \exp(\min(s_t, s_{t-1}) - \max(s_t, s_{t-1}))) + \max(s_t, s_{t-1})$
    $hmap$.put(key, s)
  return s

threshold. For each query, the response is 1 if the true response plus $Laplace(4/\varepsilon')$ is greater than $Laplace(2/\varepsilon')$ plus the threshold. Each call to AboveThreshold is guaranteed to halt only if it reaches a positive query. AboveThreshold is ε'-differentially private.

The Sparse algorithm makes T calls to AboveThreshold with a privacy parameter $\varepsilon' = \frac{\varepsilon}{T}$, where T is the number of required positive queries (features to be selected). Each time a positive query is reported, the algorithm restarts the remaining stream of queries on a new instantiation of AboveThreshold. Sparse halts after it has found the required number of positive queries T. The Sparse algorithm is ε-differentially private, following from the composition Theorem 2.3.2.

The Sparse algorithm has the same privacy guarantee as Algorithm 5 when the feature quality metric is the count heuristic. While both techniques are similar in their privacy efficiency, Algorithm 5 is easier to understand, does not require a complicated proof, and is generalizable to new feature quality heuristics. For the Sparse algorithm, the threshold needs to be fixed in advance, which causes inconvenience. Thus we believe that Algorithm 5 is a useful contribution from a practical perspective.

Multiplicative Weights Exponential Mechanism (Hardt and Rothblum, 2010) is another technique for handling large collections of queries. It can be thought of as a boosted version of the exponential mechanism which maintains an approximating distribution over the domain of possible data samples. At each iteration of the algorithm, it selects a query with high error via the exponential mechanism with a quality score the difference between the true response of the query and the response on the approximating dataset. The algorithm then computes a noisy version of the response on the true dataset using the Laplace mechanism. Finally, the approximating distribution is adjusted using the multiplicative weights rule so that it performs better on that query.

The privacy proof is as follows, assuming that all queries are 1-sensitive. If we run T iterations of MWEM by the composition theorem, it is sufficient to compute the privacy budget consumed by each iteration. At each iteration, the exponential mechanism and the Laplace mechanism are both invoked with privacy parameter $\varepsilon/2T$. By sequential composition, Theorem 2.3.2, the whole algorithm is private with parameter $2T \times \varepsilon/2T = \varepsilon$.

MWEM focuses on a few of the queries and tries to improve the response to them as well as it can. Using this algorithm for feature selection, the question becomes whether it is better to expend a fraction of the privacy budget on determining which features to evaluate accurately, then providing noisy measurements for all features. It seems

that MWEM might be useful for feature selection in datasets where some features are significantly more informative than others or where features are highly correlated. If features are highly correlated, then improving the measurement for some features will result in improvement in the measurements for the rest of the features. In order to use MWEM for feature selection, in the worst case we would need to perform D linear queries, where D is the dimensionality of the original dataset, and thus we would lose privacy budget compared with Algorithm 5.

Another issue with MWEM is that it is computationally inefficient: it manipulates a probability distribution over a set exponentially large in the dimension of the data space. This makes the algorithm difficult to scale to data of more than a few tens of features.

DualQuery (Gaboardi et al., 2014) also generates synthetic data. It differs from MWEM in that it has improved computational scalability to higher-dimensional data. We suggest that DualQuery is explored in future work on private feature selection, as the benefits of generating synthetic data and selecting features based on it remained unclear compared to selecting features directly.

Maximum Likelihood Postprocessing for Differential Privacy under Consistency Constraints (Lee et al., 2015) also addresses the problem of differentially private data release. Similar to other approaches, it shows that the accuracy of many data queries can be improved by post-processing the perturbed data to ensure consistency constraints that are known to hold for the original data. They formulate the post-processing approach as a constrained maximum likelihood estimation problem equivalent to constrained l_1 minimization. This and similar works are orthogonal to the approach taken in this thesis, but we believe that investigation into synthetic data release prior to applying machine learning methods could be a useful contribution. In particular, it would be useful to investigate the implications of the principle of least information (Dwork and Roth, 2014) in more detail.

3.6 Experiments

A major goal of this work has been to provide a differentially private classifier whose performance scales well with the number of features in a dataset. The principle of minimum information (Dwork and Roth, 2014) lead to the design of a private Naive Bayes classifier, due to its efficiency in number of parameters and training. Unfortunately, even with this most straightforward of fitting procedures, the private Naive Bayes classifier has not performed well on real datasets of high dimensionality.

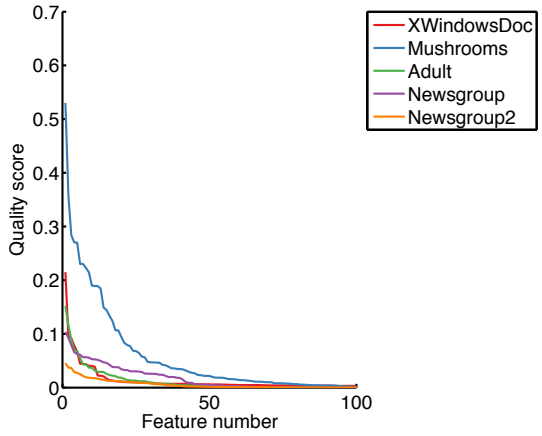


Figure 3.13: The mutual information, measured in bits, of the top hundred most informative features for the first five UCI datasets (XWindowsDoc through Newsgroup2).

A natural step towards improving this mechanism's scalability is dimensionality reduction. We have proposed two heuristics for the quality of features, based on the mutual information and feature counts; as well as a procedure for feature selection based on these heuristics.

With the ability to privately obtain a subset of features, it is now time to compare the proposed heuristics. In more detail, experiments will be conducted using the private feature sampling algorithm, Algorithm 5, with each of the two heuristics. Each of the selected feature subsets is used to train a $\varepsilon/2$-differentially private Naive Bayes classifier. These two classifiers are compared against another private Naive Bayes classifier trained on a non-privately selected feature subset.

Data We use six real world datasets, the details of which can be found in Appendix A. The datasets have been chosen to mimic different settings which are often encountered in practice: XWindowsDoc dataset mimics a large D small N problem; the Adult, Newsgroup2 and Newsgroup3 represent imbalanced problems; the Mushrooms and Newsgroup datasets exemplify moderate sized datasets with easy to handle dimensionality. All datasets are or have been converted into binary data (more details are available in Appendix A). Most importantly, numeric features have been discretized and categorical features have been one-hot encoded. Features with missing data have been removed.

Procedures compared The experiments compare several different potential ways to train a private NB classifier. First, a private NB classifier is trained using all features, i.e. the original data. Second, feature selection is employed prior to training with the goal of dimensionality reduction. The effects on private Naive Bayes's accuracy due to three different feature selection approaches are examined: non-private feature selection,

private feature selection with a quality score function based on the mutual information of a feature and the class, and private feature selection with the count-based quality score function. Finally, these results are compared against the majority baseline, as well as a non-private Naive Bayes classifier. All classifiers are 1-Laplace smoothed and trained by maximum likelihood.

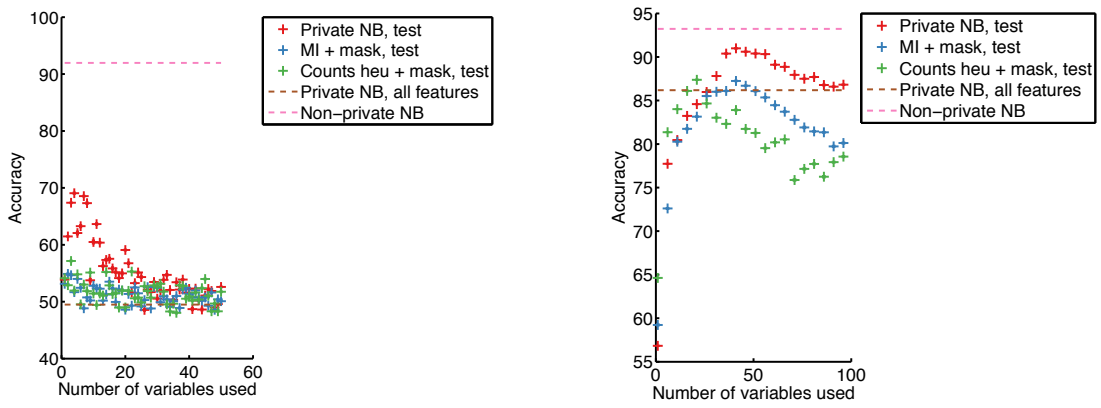
Data partitioning All of the available training data is used at each step – feature selection and training. The privacy level of the resulting procedures is controlled by the privacy parameter and the sequential composition theorem 2.3.2. An alternative approach could have split the data into disjoint subsets and used one subset per step. Differential privacy would then have had to be proven by the parallel composition theorem 2.3.3.

For a Naive Bayes classifier, the relationship between the accuracy and ε is not significantly different from the relationship between the accuracy and the dataset size. This means that for a certain dataset, similar performance is expected from the algorithm if ε is reduced twice and if the dataset size is reduced twice. Indeed, in the Newsgroup dataset, a 0.5-differentially private NB classifier trained on all data achieves 78.6% train accuracy; a 1-differentially private NB classifier trained on half of the available data achieves 78.4%. Thus, partitioning the dataset versus dividing the budget has little effect on accuracy.

Privacy settings The privacy level for these experiments is set to 0.1. This is a common setting of the privacy parameter, as it is small enough to simulate settings where privacy restrictions are significant, but big enough to allow limited amount of data to still be useful. Consider Figure 3.14, where two extreme settings of the privacy parameter have been tested on the Newsgroup dataset - 0.01 and 1. The plot shows the accuracies of the private Naive Bayes classifiers for the three feature selection techniques (one non-private and two private). The majority baseline for this dataset is 57%.

In the first plot, the privacy parameter is too small and the two fully 0.01-differentially private classifiers have baseline performance. Feature selection has not been able to improve upon the performance of the 0.01-differentially private classifier trained with all features.

In the second plot, the privacy is quite relaxed and the private classifier is only 10% away from the non-private in terms of accuracy. In such situations the advantages of feature selection are minimal, because the performance of the private classifiers is already quite good. In such situations it might be better to avoid feature selection and



(a) $\varepsilon = 0.01$ **(b)** $\varepsilon = 1$

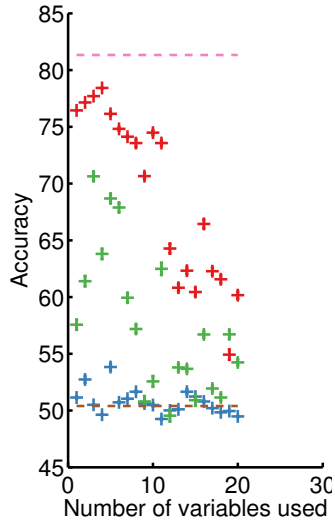
Figure 3.14: Effect on accuracy of private Naive Bayes classifiers due to changing the privacy parameter on the Newsgroup dataset (majority baseline - 57%). On the horizontal axis is the number of features used to train each classifier. Each plus mark corresponds to a classifier trained on x features. The red pluses show a 0.1-differentially private classifier with non-private feature selection according to mutual information. The blue pluses show a 0.05-differentially private classifier with 0.05-differentially private classifier with 0.05-differentially private mutual-information-based feature selection. The green pluses show a 0.05-differentially private classifier with 0.05-differentially private count-based feature selection. The green and blue training procedures are jointly 0.1-differentially private. The pink dotted line represents a non-private Naive Bayes classifier trained on all features. The purple dotted line represents a 0.1-differentially private classifier trained on all features. All plus marks correspond to results after feature selection, but only the blue and green marks represent fully-private algorithms.

save the privacy budget for training.

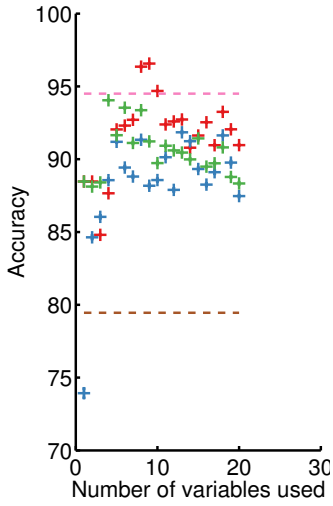
Thus, private feature selection is most likely to be useful in situations where data is limited to moderate, and where the privacy restriction is significant. These are the settings in which we compare feature selection methods.

Namely, the private classifier with no feature selection is 0.1-differentially private; the classifier with feature selection is also 0.1-differentially private, where half of the privacy budget is used for feature selection, and the other half, for training.

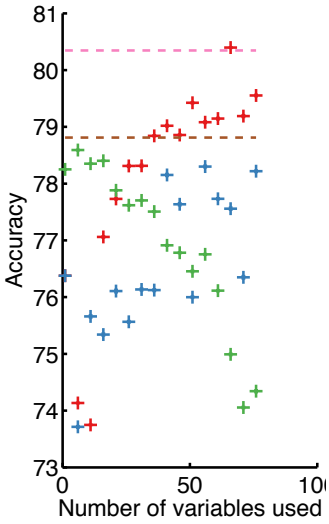
Additional settings The number of times each of the randomized procedures is repeated is 10 and the maximum number of features selected is around 60.



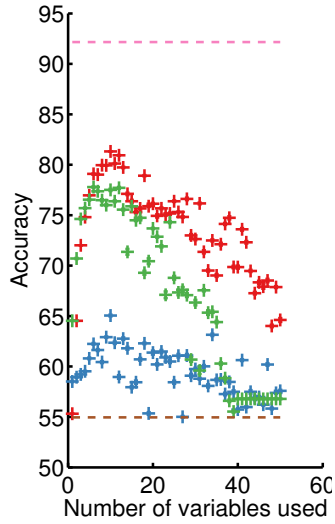
(a) XWindowsDoc dataset, majority baseline is 50%.



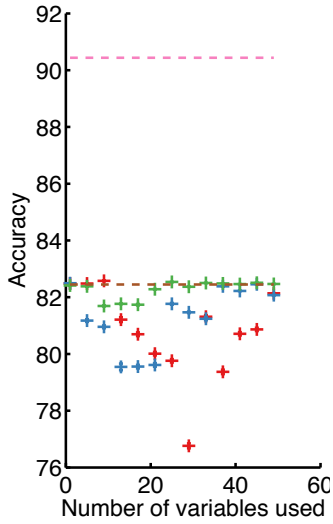
(b) Mushrooms dataset, majority baseline is 52%.



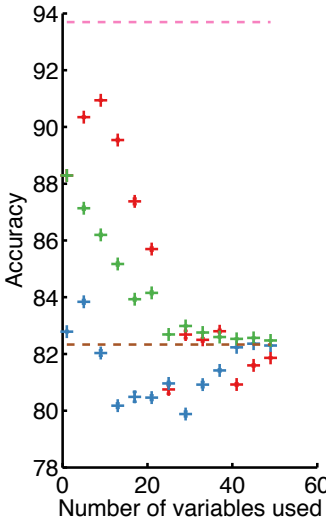
(c) Adult dataset, majority baseline is 76%.



(d) Newsgroups dataset, majority baseline is 57%.



(e) Newsgroups2 dataset, majority baseline is 81%.



(f) Newsgroups3 dataset, majority baseline is 82%.

Figure 3.15: Effect on performance gains from private feature selection. The red pluses represent a 0.1-differentially private Naive Bayes classifier, which is trained on a subset of the features chosen by non-private feature selection according to mutual information. The blue pluses represent a 0.05-differentially private classifier, trained on features selected by 0.05-differentially private mutual-information-based feature selection. The green pluses show a 0.05-differentially private classifier, trained on features selected by 0.05-differentially private count-based feature selection. The green and blue training procedures are jointly 0.1-differentially private. The pink dotted line represents a non-private Naive Bayes classifier trained on all features. The purple dotted line represents a 0.1-differentially private classifier trained on all features.

3.6.1 Results

The results are shown in Figure 3.15. The legend for this figure is the same as for Figure 3.14, but is detailed again here for self-containment.

Each marker in the plots represents the accuracy of one private Naive Bayes classifier trained with a specific number of features. The number of features is on the horizontal axis. The color of the plus marks represents the algorithm used for feature selection. Red stands for non-private feature selection based on the mutual information. Blue stands for 0.05-differentially private feature selection based on the mutual information. Green stands for 0.05-differentially private feature selection based on the counts heuristic. The red classifier is a 0.1-differentially private Naive Bayes; however this guarantee is broken by the non-private feature selection performed prior to training. The other two training procedures are 0.05-differentially private, making the classifiers fully 0.1-differentially private. The pink dotted line represents a non-private Naive Bayes classifier trained on all features. The purple dotted line represents a 0.1-differentially private classifier trained on all features. To sum up, all plus marks correspond to results after feature selection, but only the blue and green marks represent fully-private algorithms.

To begin with, we look at the XWindowsDoc and Newsgroup datasets. These datasets are balanced, the majority class label being 50% and 57% of cases, respectively. As obvious from the red marks in figures 3.15a and 3.15d non-private feature selection was able to improve the performance of the private classifiers. Private Naive Bayes with private mutual-information-based feature selection, in blue, produced similar results to the classifier trained on all features. Private count-based feature selection, however, did have a positive effect – accuracy of private Naive Bayes improved by around 20%, close to that achieved by non-private feature selection.

Secondly, consider the results for the Mushrooms dataset, shown in 3.15b. The dataset is balanced with a majority class share of 52%. On this dataset the private classifier trained on all features has non-trivial performance. Both the mutual information and the count-based feature selection improve that performance further, by about 10%. The effect of the two feature selection heuristics is very similar.

Let us consider the effect of using the mutual information versus the count-based feature selection on the XWindowsDoc and Mushrooms datasets, and find out why the count-based heuristic resulted in a more significant gain compared with the mutual information one on the former dataset. Figure 3.18 shows the probability of a feature

being chosen as a function of its quality score. For the XWindowsDoc dataset using the MI results in a very flat distribution over the features, whereas the count-based heuristic does not. On the Mushrooms dataset, the difference between the distributions resulting from the two heuristics is almost non-present. To understand why this is, consider Figure 3.13, which shows the mutual information for the 100 most informative features in all datasets. The values of the mutual information for the Mushrooms dataset are significantly bigger than those for the XWindowsDoc dataset. For the latter, the sensitivity of the mutual information is too big compared with the range of values, causing the signal to be trumped by the noise.

Thirdly, on Newsgroup2, Figure 3.15e, no improvement was achieved by employing feature selection, be it private or non-private. For this dataset, throwing away some features does not improve the accuracy, because there are no features which are especially useful compared with the rest. This is shown in Figure 3.13, where the most informative feature has mutual information of around only 0.05 bits.

For the Adult dataset, Figure 3.15c, non-private feature selection provides a very slight gain in accuracy over the private Naive Bayes classifier trained on all features. However, neither of the private feature selection techniques were able to achieve a similar result.

Consider the private mutual-information-based feature selection. Figure 3.13 shows that some features in the Adult dataset are more informative than others. Furthermore, Figure 3.18e, which shows the probability of a feature being selected as a function of its mutual information, suggests that the private selection procedure was fairly confident. The private feature selection chose the features that were believed to be useful. Unfortunately, it left too little privacy budget for training, meaning that the gain from the dimensionality reduction was destroyed. In Figure 3.17 we increase the privacy level to 0.4. For the Adult dataset employing feature selection is counterproductive: it is better to use the whole privacy budget and all features. The amount of data is enough to allow the private classifier to overcome slightly more relaxed privacy restrictions easily, without the need for dimensionality reduction.

With respect to the count-based heuristic, in Figure 3.18f we confirm an important observation: the heuristic is useful when there is not enough data available or the privacy constraints are extremely stringent. In other cases, it is sufficient to use the mutual information for feature selection. In more detail, because the Adult dataset is relatively big, the counts too are big. This means that the small privacy parameter does not affect the feature selection measurably. What we are seeing is akin to overfitting – the

selection procedure has become too confident that certain features are good. However, as we can see from both Figures 3.15c and 3.17, the heuristic can fail to pick the best features.

Finally, the experiment on Newsgroup3 dataset demonstrates the usefulness of the counts heuristic on an imbalanced dataset. Contrary to Adult and Newsgroup2, non-private feature selection does help here.

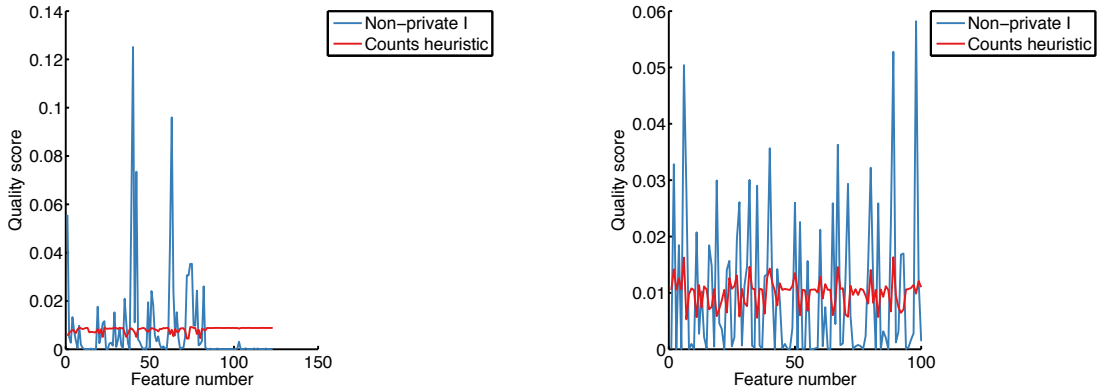
Consider the mutual information for imbalanced datasets. The maximum of the mutual information is $H_2(p(y=1))$ and this is very small when one of the classes is very small, i.e. $p(y=1)$ is close to 0 or 1. As seen previously, when the range of the mutual information is small, and it can result in flatter distribution over variables. The count-based feature selection doesn't suffer from this problem and achieves an improvement on this dataset.

In summary, private feature selection is a beneficial preprocessing step in training a private Naive Bayes classifier, when some features in the dataset are believed to be more informative than others. The count-based heuristic is especially useful in datasets where the range of the mutual information is too small, the privacy restrictions are severe, or the dataset size is small. On the other hand, if the mutual information is relatively big, the privacy constraints are lenient, or data is abundant, the mutual information can be an effective tool in feature selection.

In conclusion, we have shown that private dimensionality reduction through feature selection can make private classification practical, where it was not before. This is especially true for datasets which are high-dimensional but contain redundant features. Two heuristics for feature selection were examined. The first is based on counts of times a feature appears in the two classes, and the second is based on the mutual information of a feature and the class label. Both viable options for feature selection, the first heuristic is preferred whenever privacy is high and dataset size is small.

3.7 Conclusion

In this chapter we have developed a differentially private algorithm for training a Naive Bayes classifier. Using several real world datasets, we have demonstrated that the classifier can achieve non-trivial performance on datasets with moderate dimensionality. For problems where data is not enough, we have proposed two dimensionality reduction techniques based on feature selection. We have proved differential privacy for each of the feature selection methods. We have proven empirically that one of the techniques,



(a) Adult dataset.

(b) Newsgroup dataset.

Figure 3.16: Distribution over features resulting from the mutual information and the counts heuristic. In blue is the normalized mutual information score and in red is the normalized count score. On the horizontal axis is the feature number. A bigger score for a particular feature number means the corresponding feature is more useful. Aim of figure is to show how the Adult dataset differs from the other datasets in a way which explains the failure of the count-based heuristic on this dataset.

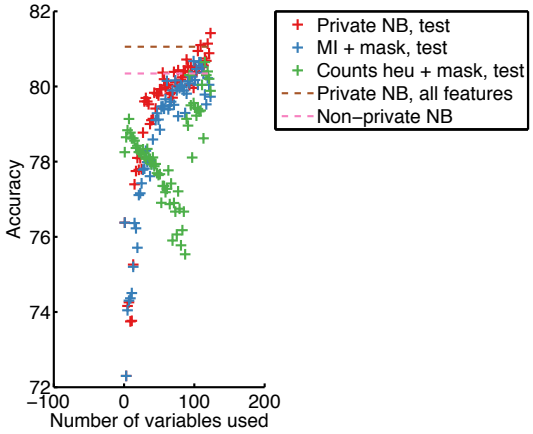
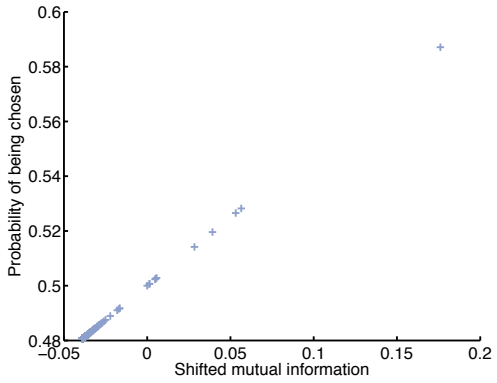
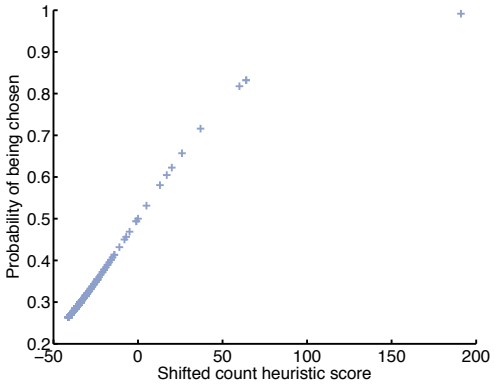


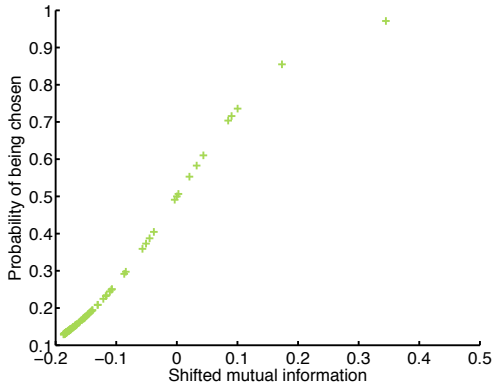
Figure 3.17: Comparing feature selection heuristics on the Adult dataset with $\varepsilon = 0.4$.



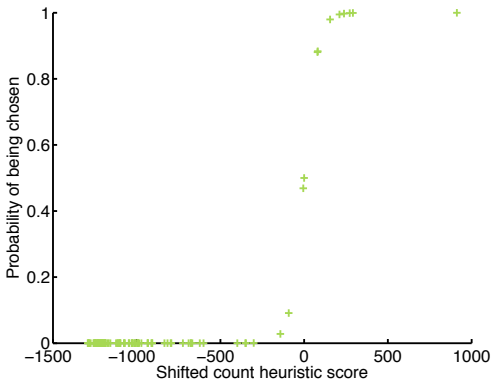
(a) XWindowsDoc dataset, mutual information.



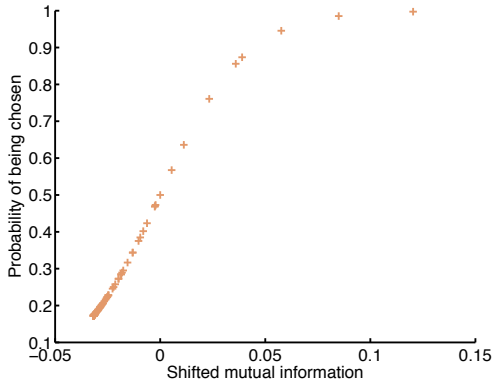
(b) XWindowsDoc dataset, counts.



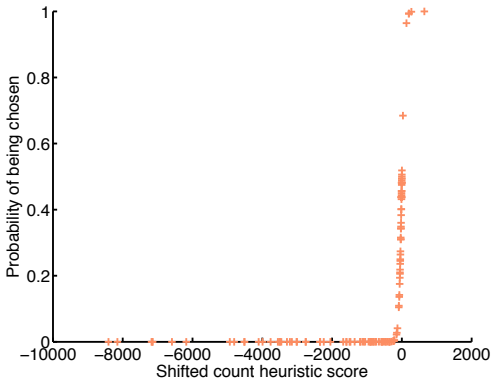
(c) Mushrooms dataset, mutual information.



(d) Mushrooms dataset, counts.



(e) Adult dataset, mutual information.



(f) Adult dataset, counts.

Figure 3.18: Mutual information versus counts heuristic score Plots show the probability of a feature being chosen as a function of its heuristic score. The bigger the slope, the more confident the feature selection is. The probabilities for counts scores are further apart compared with the those for the mutual information, so the resulting distribution over features will be more peaked. We take the number of features to be selected is 10 and $\varepsilon = 0.05$ for each feature.

based on a newly proposed feature quality heuristic, is able to improve the performance of an inadequate private classifier.

Chapter 4

Private Logistic Regression

4.1 Motivation

Naive Bayes is a generative classifier: it models the joint probability of the input $\boldsymbol{x}$ and the class label y, $p(\boldsymbol{x}, y)$, uses the Bayes rule to calculate $p(y|\boldsymbol{x})$, and picks the most likely label. A natural question is whether it is a good idea to model the joint probability, when only the posterior probability over the class is necessary for classification. The class conditional densities may contain a lot of structure that has little effect on the posterior probabilities of the class labels. Approaches that model the posterior probabilities of class labels directly are called discriminative models. On the one hand, a classifier should not solve a more general problem as an intermediate step to the classification problem, but on the other, there is no general criterion for choosing between discriminative and generative classifiers (Ng and Jordan, 2002).

Clearly, it is important to examine both generative and discriminative classifiers under differential privacy. We investigate logistic regression, since it is the discriminative equivalent of Naive Bayes for discrete data.

We begin by an extensive review of the main private l_2-regularized logistic regression mechanisms, followed by an investigation of their accuracy on the real datasets previously used. The chapter concludes with empirical evidence suggesting that feature selection is unlikely to improve the performance of private logistic regression.

4.2 Logistic Regression

Logistic regression is a discriminative classifier which is linear in the parameters. The logistic regression model is as follows:

$$p(y|\boldsymbol{x},\boldsymbol{\theta}) = \text{Ber}(y|\text{sigm}(\boldsymbol{\theta}^T\boldsymbol{x})). \tag{4.1}$$

To fit the logistic regression model parameters, we could minimize the normalized negative log-likelihood of the parameters given the data:

$$NLL(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \mu_n + (1-y_n)\log(1-\mu_n)\right], \tag{4.2}$$

where $\mu(\boldsymbol{x}) = \text{sigm}(\theta^T\boldsymbol{x}) = p(y=1|\boldsymbol{x})$. To avoid overfitting, the following regularized objective function is used: $NLL(\boldsymbol{\theta}) + \frac{\lambda}{2N}\|\theta\|_2^2$. This objective function is minimized by an iterative gradient-based optimization algorithm, because the solution cannot be found analytically.

The most famous differentially private algorithm for logistic regression was designed in the context of empirical risk minimization (Chaudhuri et al., 2011). In that context, the objective function of logistic regression can be rewritten as:

$$J(\theta,\mathcal{X}) = \frac{1}{N}\sum_{n=1}^{N} l(h(x^{(n)}),y^{(n)}) + \frac{\lambda}{2N}\|\theta\|^2. \tag{4.3}$$

In the following we will refer to l as the loss function and to the term $\frac{\lambda}{2N}\|\theta\|^2$, as the regularizer.

We state one interpretation of the regularization parameter λ, as it is useful later on. The posterior probability of the model parameters given the data is proportional to the likelihood of the parameters times the prior:

$$p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta)p(\theta). \tag{4.4}$$

Taking the log the above becomes

$$\log p(\theta|\mathcal{X}) = \log p(\mathcal{X}|\theta) + \log p(\theta) + const. \tag{4.5}$$

If the model parameters are assumed to be Gaussian, $p(\theta) = \mathcal{N}(\theta;0,\sigma^2)$, then

$$\log p(\theta) = -\frac{1}{2}\frac{\theta^2}{\sigma^2} + const. \tag{4.6}$$

Finally we can set the regularization parameter $\lambda = \frac{1}{\sigma^2}$. Rewriting this in terms of the standard deviation σ, we conclude that $\sigma = \sqrt{\frac{1}{\lambda}}$.

We now review the most established private logistic regression models in the literature.

4.3 Output perturbation

The most straightforward method of designing a private logistic regression mechanism is output perturbation. Generally speaking, output perturbation involves computing the vector of interest using a non-private mechanism and only then adding privacy-related noise before finally outputting the vector. In the case of logistic regression, this involves optimizing the logistic regression objective function to compute a parameter vector; then drawing a noise vector from a certain parametric family of distributions with zero mean and some variance; and finally releasing the sum of the parameter vector and the noise vector (Chaudhuri et al., 2011).

With respect to the sensitivity of the objective function, under certain assumptions on the data, regularizer and loss function, the following theorem holds (Chaudhuri et al., 2011):

Theorem 4.3.1. For a differentiable and 1-strongly convex regularizer, and a convex and differentiable loss function with $\|l'(z)\| \leq 1$ for all z, the sensitivity of the cost function is at most $\frac{2}{\lambda}$, where λ is the regularization constant.

Applying the theorem to logistic regression relies on the data being preprocessed such that the norm of each sample is bounded by 1, i.e. $\|\boldsymbol{x_n}\|_2 \leq 1$. As a consequence, the noise vector $\boldsymbol{b}$ is sampled from a distribution with scale $\frac{\Delta}{\varepsilon} = \frac{2}{\lambda\varepsilon}$. The algorithm is detailed in 6.

Algorithm 6 Output perturbation version of private logistic regression

1: **procedure** PRIVATEERMOUTPUTPERT($X, y, \varepsilon, \lambda$) ▷ Data X, privacy parameter ε, regularization constant λ
2: Draw a noise vector $\boldsymbol{b}$ according to $p(\boldsymbol{b}) \propto \exp\left(-\frac{\lambda\varepsilon}{2}\|\boldsymbol{b}\|_2\right)$.
3: **return** $\theta_{priv} = \underset{\theta}{\operatorname{argmin}}\, J(\theta, X) + \boldsymbol{b}$

When fitting logistic regression, cross-validation is usually performed to find the best regularization parameter. Private logistic regression is no exception. Consider the result of cross-validation with the output mechanism for logistic regression. The mechanism adds noise with scale $\frac{2}{\lambda\varepsilon}$, so the standard deviation of the noise distribution is $\frac{2\sqrt{2}}{\lambda\varepsilon}$. If λ is very big, then the scale of the privacy noise will be close to zero and the performance of the classifier will be close to that of the non-private one. Recall further the relationship between the regularization parameter and the variance of the prior on

the model parameters:

$$\sigma = \sqrt{\frac{1}{\lambda}}. \tag{4.7}$$

Both the privacy noise distribution and the distribution on the possible model parameters depend on the value of the regularization parameter. For a big regularization parameter, the privacy noise will be negligible, but the model parameters might be forced too close to 0 and the classifier might underfit the data. For small values of λ, the privacy noise will be bigger, but the model will be more complex, since the learned weights will be bigger. There is an important difference between how λ affects the two distributions: the standard deviation of distribution of the weights will reduce with $\sqrt{\lambda}$, whereas the deviation of the privacy noise will reduce with λ. Therefore, cross-validation will prefer bigger values of λ.

Next we describe one of the most interesting differentially private methods of the past few years.

4.4 Objective perturbation

Objective perturbation was first proposed by (Chaudhuri and Monteleoni, 2008) and (Chaudhuri et al., 2011) as a technique for creating differentially private classifiers learned via empirical risk minimization. In this text the term Private ERM is used for brevity to refer to private logistic regression by objective perturbation.

The objective perturbation method flowed out of an understanding of the mutually beneficial relationship between regularization and differential privacy. The authors argue that since regularization is used to prevent the extreme reliance of the learned parameters on the data, it should also partially help satisfy differential privacy.

The main idea behind objective perturbation is to add noise to the objective function itself and only then to optimize it.

4.4.1 Algorithm

For a dataset $\mathcal{X}$ containing N samples and a cost function $J(\theta, \mathcal{X})$, the private version of J is shown below. It takes as inputs the training data $\mathcal{X}$, the privacy parameter ε, the regularization constant λ, and a constant c.

$$J_{priv}(\boldsymbol{\theta}, \mathcal{X}) = J(\boldsymbol{\theta}, \mathcal{X}) + \frac{\boldsymbol{b}^T \boldsymbol{\theta}}{N}, \tag{4.8}$$

where $J(\boldsymbol{\theta}, \mathcal{X})$ is the standard regularized logistic regression objective. In (4.8) $\boldsymbol{b}$ is a random noise vector with density

$$p(\boldsymbol{b}) = \frac{1}{\alpha} e^{-\beta\|\boldsymbol{b}\|} \tag{4.9}$$

where α is a normalizing constant and β is a function of the privacy parameter ε and the sensitivity of the cost function J.

We now present the algorithm and delve into its specifics. From (Chaudhuri et al., 2011):

Algorithm 7 Objective perturbation

procedure PRIVATEERM($X, y, \varepsilon, \lambda, c$) ▷ Data X, privacy parameter ε, regularization constant λ
  $\varepsilon' = \varepsilon - \log(1 + \frac{2c}{\lambda} + \frac{c^2}{\lambda^2})$
  if $\varepsilon' > 0$ **then**
    $\Delta = 0$
  else
    $\Delta = \frac{c}{N(e^{\varepsilon/4}-1)} - \frac{\lambda}{N}$
    $\varepsilon' = \frac{\varepsilon}{2}$
  $\beta = \frac{\varepsilon'}{2}$
  Draw a vector $\boldsymbol{b}$ according to 4.9
  return $\theta_{priv} = \underset{\theta}{\operatorname{argmin}}\, J(\theta, \mathcal{X})_{priv} + \frac{1}{2}\Delta\|\theta\|^2$

The original presentation of the algorithm uses $\Lambda = \frac{\lambda}{N}$. We chose to use small λ in the presentation of the algorithm and experiments, because it is more natural when the average loss is being optimized.

The constant c is chosen analytically so that the following theorem is satisfied:

Theorem 4.4.1. If the regularizer is 1-strongly convex and doubly differentiable, and the loss $l(\boldsymbol{z})$ is convex and doubly differentiable, with $\|l'(z)\| \leq 1$ and $\|l''(z)\| \leq c$, $\forall z$, then the objective perturbation algorithm preserves ε-differential privacy.

For logistic regression $c = \frac{1}{4}$, since $l''(z) = \frac{1}{(1+e^{-z})(1+e^{z})}$. For other algorithms, it would be necessary to find the upper bound of the second derivative of the loss function in order to determine c.

The proof of Theorem 4.4.1 relies on the sensitivity of the cost function, see Theorem 4.3.1. Importantly, both of these theorems assume that the norm of each sample is

bounded by 1. One way of achieving this is to use the maximum of feature values. Computing this statistic without spending privacy budget may break the guarantee of the complete learning procedure. Since this issue is not addressed in (Chaudhuri et al., 2011), we leave it as future work and recommend that care is taken in real-world applications.

4.4.2 Noise behaviour for large N

For all settings of the privacy parameter ε, the variance of the noise vector $\boldsymbol{b}$ in the algorithm is constant with respect to the number of samples N. Thus the variance of the noise term $\frac{\boldsymbol{b}^T\boldsymbol{\theta}}{N}$ goes to zero with large N, and the cost function tends to that of the non-private algorithm.

4.4.3 Noise behavior with D

The algorithm adds a scaled dot product of a D-dimensional noise vector and the weight vector to the non-private objective function. The result is a linear shift to the objective. The new position of the optimum after this perturbation depends on the shape of the objective function. If the objective is very flat, a small perturbation to it can result in a large shift of the optimal parameters. However, if the objective is strongly peaked, the shift to the optimum will be very small.

The scale of the noise is independent of the data dimensionality D. However, for problems where the dimensionality is high and data is not enough, the objective function may be fairly flat, which will result in big changes to the optimal parameters after objective perturbation.

4.4.4 Interpretation and comparison with output perturbation

Consider the values for which the algorithm prefers the first branch, $\varepsilon' = \varepsilon - \log(1 + \frac{2c}{\lambda} + \frac{c^2}{\lambda^2}) \geq 0$.

$$\log(1+\frac{2c}{\lambda}+\frac{c^2}{\lambda^2}) \approx \frac{2c}{\lambda} \text{ for large } \lambda \tag{4.10}$$

$$\implies \varepsilon' \approx \varepsilon - \frac{2c}{\lambda} > 0 \tag{4.11}$$

$$\iff \lambda > \frac{1}{2\varepsilon} \tag{4.12}$$

ε	$\varepsilon', branch1$	$\varepsilon', branch2$
$\varepsilon = 0.1$	$\lambda \geq 5$	$\lambda \in (-0.5, 5)$
$\varepsilon = 0.5$	$\lambda \geq 1$	$\lambda \in (-0.2, 1)$
$\varepsilon = 1$	$\lambda \geq 0.5$	$\lambda \in (-0.25, 0.5)$
$\varepsilon = 10$	$\lambda \geq 0.005$	$\lambda \in (-0.0025, 0.005)$

Table 4.1: Example values for λ and ε', showing when a branch is preferred.

Therefore, the algorithm opts for the first branch if $\lambda > 1/2\varepsilon$. Table 4.1 shows some example values for λ and ε. The negative intervals are ignored, as it is common to have a positive λ. We see that in the first branch of the conditional we use more regularization, so we can afford to use a less restrictive privacy budget. Conversely, in the second branch the smaller amount of regularization is not enough to achieve privacy and this is reflected in the tighter privacy budget.

The algorithm looks at λ and makes a choice depending on whether it is big or small. When λ is sufficiently big, the scale of the noise vector distribution is roughly $\frac{4\lambda}{2\lambda\varepsilon - 1}$. For a small λ (very close to zero), the algorithm discards the original regularization constant and picks one that is 10 when ε is 0.1, 100 when ε is 0.01, 1000 when ε is 0.001, and so on. The scale of the noise distribution is $\frac{4}{\varepsilon}$.

In cross-validation, will the objective perturbation algorithm prefer a big or small regularization parameter? We can find the answer to this question by considering the distribution over model parameters and the noise distribution. Consider the following example, where $\lambda_1 = 0.1$ and $\varepsilon = 0.1$. The algorithm will pick the second branch and make a new regularization parameter equal to 10; the standard deviation of $p(\theta)$ will be $\frac{1}{\sqrt{\lambda}} = \frac{1}{\sqrt{10}}$; the standard deviation of the noise will be $\frac{4\sqrt{2}}{\varepsilon} = 40\sqrt{2}$. Now take $\lambda_2 = 10$. The first branch will be chosen; the standard deviation of the weights will be again $\frac{1}{\sqrt{10}}$; the deviation of the noise, $\frac{4\lambda}{2\lambda\varepsilon - 1} = 40\sqrt{2}$. The two branches will have the same effect and the performance of the algorithm will be identical. Whatever the algorithm chooses, the effective regularization will be significant.

We now compare, in the context of $\varepsilon = 0.1$, the objective perturbation with the output perturbation algorithm above. Since both mechanisms have large effective regularization, it suffices to compare them for large values of λ.

The objective perturbation method will add noise with scale $\frac{4\lambda}{2\lambda\varepsilon - 1}$ or $\frac{4}{\varepsilon}$. The output perturbation method adds noise with scale $\frac{2}{\lambda\varepsilon}$. For $\varepsilon = 0.1$ and $\lambda > 1$ the scale of the output perturbation is always smaller than the scale of the objective perturbation noise.

If the objective function is very flat, it is possible that the optimum is perturbed a lot as a result of the perturbation of the objective function. In such situations, because the scale of the output perturbation noise is smaller, the objective perturbation method may perform worse than the output perturbation method. However, if the objective function is very peaked, it is possible that the objective perturbation method achieves better results.

Table 4.2 shows the average test accuracies achieved by output perturbation and objective perturbation on several datasets, detailed in Table 3.1. The privacy level is set to $\varepsilon = 0.1$ and each procedure is run 10 times. Regularization parameters are found by 5-fold cross-validation, where $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$. Both private mechanisms for logistic regression perform poorly compared with standard logistic regression, most often than not achieving baseline results. The output perturbation mechanism performs better than objective perturbation on two datasets, suggesting that the objective perturbation is unlikely to be a significant improvement upon output perturbation in real-world applications. Table 4.2 suggests that both algorithms for private logistic regression will have trouble producing useful results on real-world data.

Following the positive effect of dimensionality reduction on the accuracy of private Naive Bayes classifiers, we now go on to investigate feature selection with private logistic regression. Even though our results suggest that objective perturbation may not necessarily be the clear winner in practical applications, we perform further experiments with this algorithm, since it is claimed to have slightly better theoretical guarantees, in particular, better sample complexity (Chaudhuri et al., 2011). Additionally, Private ERM (Chaudhuri et al., 2011) has attracted some attention in the research community and remains an often cited paper.

4.5 Empirical analysis

In the following section we evaluate logistic regression under differential privacy as represented by the Private ERM algorithm (Chaudhuri et al., 2011). The first goal is to test the usefulness of private logistic regression compared with the private Naive Bayes classifier on real datasets. In the non-private setting, logistic regression usually outperforms Naive Bayes classifiers, because logistic regression optimizes an objective function that directly relates to classification performance. In the private setting, the performance of the two algorithms depends on the sensitivity and the scale of the noise they incur. A parallel between private logistic regression and private Naive Bayes cannot

Dataset	Logistic regression	Objective perturbation	Output perturbation	Majority baseline
XWindowsDoc	80 (3)	50 (-3)	73 (-3)	50
Mushrooms	100 (2)	50 (2)	71 (3)	52
Adult	85 (2)	76 (3)	76 (4)	76
Newsgroup	91 (0)	56 (-2)	61 (4)	57
Newsgroup2	90 (1)	81 (2)	81 (3)	81
Newsgroup3	93 (-1)	82 (5)	82 (4)	82

Table 4.2: Comparison between the output perturbation method and the objective perturbation method for private logistic regression. Shown are the test accuracies (in percent, rounded to the nearest digit) over 10 runs of the algorithms for $\varepsilon = 0.1$, where λ was chosen by 5-fold cross-validation. In the brackets, we show the exponent of the most often picked regularization parameter, i.e. the mode of $\log_{10}\lambda$ over all runs. The set of possible regularizers is $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$. The two methods produce very similar results and there is no evidence to suggest that the objective perturbation method is generally superior in practice. The final column shows the majority baseline for each dataset, i.e. the prediction accuracy of a classifier which always predicts the bigger class. The results suggest that both algorithms for private logistic regression will have trouble producing useful results on real-world data.

be drawn solely from the discriminative nature of the former and empirical investigation is needed.

4.5.1 Experimental setup

Data Experiments are performed with the Private ERM algorithm on the six datasets used previously in Private Bernoulli Naive Bayes MLE based on the Laplace mechanism: XWindowsDoc, Mushrooms, Adult, and Newsgroup 1, 2, and 3, detailed in Appendix A.

The datasets have been chosen to mimic different settings which are often encountered in practice: XWindowsDoc dataset mimics a large D small N problem; the Adult, Newsgroup2 and Newsgroup3 represent imbalanced problems; the Mushrooms and Newsgroup datasets exemplify moderate sized datasets with easy to handle dimensionality. All datasets are or have been converted into binary data. The exact preprocessing done for each dataset is detailed in Appendix A. Most importantly, numeric features have been discretized and categorical features have been one-hot encoded. Features with missing data have been removed.

Procedures compared The following procedures were compared. First, a private logistic regression classifier is trained using all features, i.e. the original data. Second, feature selection is employed prior to training with the goal of dimensionality reduction. The effects of three different approaches to feature selection are examined: non-private feature selection, private feature selection with a quality score function based on the mutual information of a feature and the class, and private feature selection with the count-based quality score function. Finally, these results are compared against the majority baseline, defined as the proportion of samples in the bigger class, a non-private logistic regression, and a Naive Bayes classifier.

Cross-validation The non-private NB classifier is 1-Laplace smoothed and the logistic regression is trained by MAP.

Importantly, for all logistic regression classifiers, the regularization parameter was chosen by *non-private* 5-fold cross-validation. The complete set of regularization parameters considered is $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$. Note that non-private cross-validation breaks the formal privacy guarantee, since the composition theorem is violated. Future work should attempt to correct this shortcoming.

Budget partitioning As always, when employing an additional pre-training step, we have to ensure that this step solidifies rather than breaks the end-to-end privacy of

the training procedure. As previously explained, one way to achieve this is to dedicate a portion of the available privacy budget. An alternative is to partition the original dataset into two disjoint datasets and use each one for the separate steps.

We have seen that for private Naive Bayes there is no clear advantage of either approach when the data is moderate and when the privacy budget is relatively low. Private logistic regression behaves similarly, as we will now show.

There are two cases to consider as part of the algorithm; however, as discussed previously, cross-validation will prefer the first case. In this case $\Delta = 0$ the last term disappears and the objective function is

$$J_{priv}(\theta, \mathcal{X}) = \frac{1}{N}\left(\sum_{n=1}^{N} l(h(x^{(n)}), y^{(n)}) + \frac{1}{2}\lambda\|\theta\|^2 + b^T\theta\right). \tag{4.13}$$

If the number of samples was to decrease twice, to preserve the same minimum, λ will have to increase twice and b will have to increase twice. The scale of the noise added is $\beta = 2/\varepsilon' \approx 2/\varepsilon$. If ε was decreased twice, the scale of the noise would also increase twice. The variance of the noise would increase 4 times and the standard deviation would increase twice. This would make the resulting noise vector b twice as large. Thus, halving the amount of data and halving the privacy parameter will decrease the accuracy of the algorithm similarly.

Privacy settings To allow comparison of the performance of private logistic regression with private Naive Bayes, the privacy parameter for this set of experiments is also set to $\varepsilon = 0.1$. Previous experiments with private logistic regression have shown that at this privacy level it is difficult to obtain satisfactory results unless abundant data is available (cf. Appendix B). When feature selection is employed, the privacy budget is equally split between feature selection and training.

Additional settings Each randomized procedure is repeated 10 times. The vertical lines show the standard error, computed by dividing the standard deviation of the accuracies for each feature subset by the square root of 10. Due to the low performance of private logistic regression the standard deviations are small and so the vertical lines are sometimes not clearly visible in the plots. The optimization routine used is minimize.m (Rasmussen, 2014).

4.6 Results and Discussion

Results are shown in Figure 4.2, and the legend is shown in Figure 4.1. For all datasets non-private logistic regression matches or outdoes the performance of the Naive Bayes

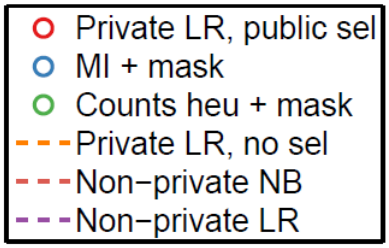


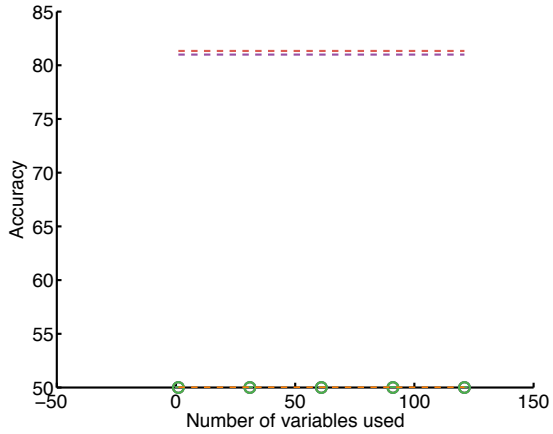
Figure 4.1: Legend for Figure 4.2. Each empty circle represents the mean test accuracy for a particular subset of the features in a dataset and the vertical lines show the standard error. The orange dashed line refers to private logistic regression with no feature selection. Red stands for the private logistic regression with non-private feature selection by the mutual information with the class; blue stands for private logistic regression with private feature selection using the mutual information; green stands for private logistic regression with private feature selection using the count heuristic. Finally, the red and purple dashed lines refer to the standard Naive Bayes and logistic regression, respectively.

classifier. With privacy, the situation is much bleaker: for all datasets, private logistic performance using all features achieves close to baseline performance. Feature selection is unable to improve the accuracy to a satisfactory level.

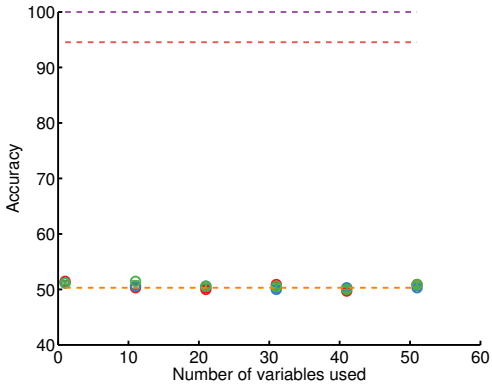
Insight into why this algorithm performs badly can be obtained by considering the difference between the regularization term and the privacy noise term. The Gaussian prior on weights helps by reducing weights of the features whose estimate is uncertain. The uncertainty is due to not having enough data or not having conclusive data. In contrast, the privacy noise term penalizes all weights, independent of how good their estimate is. In essence, the private algorithm puts a restriction on how informative a feature can be independent of how well supported it is by the data. It is difficult to see how this algorithm could be useful in practice unless there is enough data to compensate for the noise.

4.6.1 Conclusion

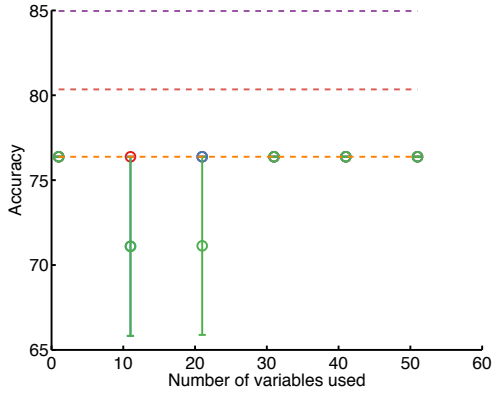
In this chapter we have compared the output and objective perturbation algorithms for private logistic regression, showing that their performance is mostly unsatisfactory and may be problem specific. We have concluded that in practice, cross-validation will result in both algorithms regularizing the objective function significantly. Finally, we have demonstrated that for the objective perturbation algorithm private feature selection



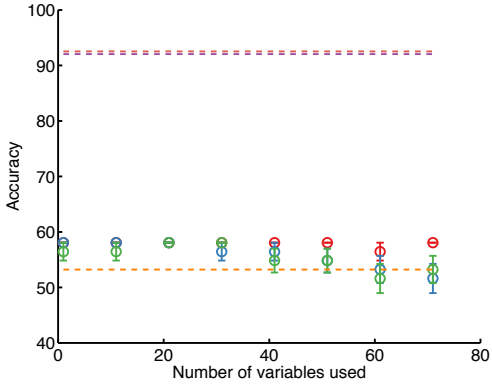
(a) XWindowsDoc dataset.



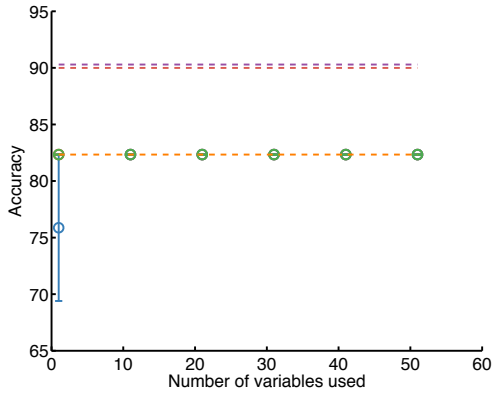
(b) Mushrooms dataset.



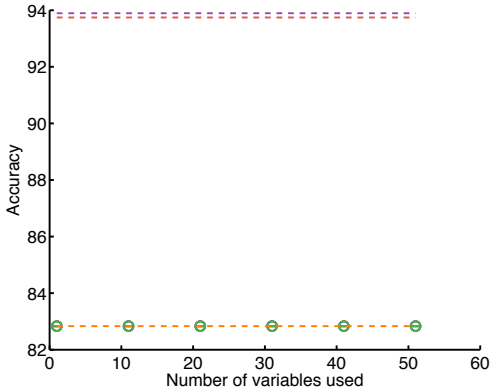
(c) Adult dataset.



(d) Newsgroups dataset.



(e) Newsgroups2 dataset.



(f) Newsgroups3 dataset. Counts heuristic helps with this imbalanced dataset.

Figure 4.2: Comparison of the count-based score function and the mutual score function, using Algorithm 7. Feature selection and training are jointly 0.1-differentially private. For each subset of features and each run of the training procedure, 5-fold cross-validation is used to select one out of the following set of regularization parameters: $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$.

is unlikely to improve performance. Performance is worse than the that of differentially private Naive Bayes and it is often close to a simple guessing baseline.

Chapter 5

Future Work

In Chapter 1 we developed a differentially private Naive Bayes classifier for binary features. A natural extension is to develop Naive Bayes classifiers for numeric and categorical data. As part of such future work, the sensitivity of the private mechanism's quality function would have to be derived. For real-valued features there is a danger that the sensitivity of the quality function could become unbounded. To avoid this problem, the features would have to be bounded by rescaling. Having derived the new sensitivity of the quality function, the private Naive Bayes fitting procedure developed in this work can be used with very little modification.

In this thesis and in previous work, feature rescaling has been assumed to be a safe data preprocessing step. In practice it is common to use certain statistics about features in a dataset, such as the minimum and maximum of features (the columns of the design matrix). However, using such statistics may not guarantee differential privacy and could potentially break the privacy guarantee of the whole learning process. Such operations may need to be made differentially private by spending certain amount of the total privacy budget and applying the differential privacy composition theorems. Further investigation into feature normalization and scaling is important future work.

A more significant improvement upon the current work comes from a change in mindset. In this thesis, and all prior work on differentially private machine learning methods, the tendency has been for a private mechanism to output private model parameters. For example, in our private Naive Bayes classifier, the Laplace mechanism was employed to add noise to the model parameters prior to their release. The goal was to allow the data holder to share the private model with an interested party such that they could make predictions about previously unseen data.

The principle of least information (Dwork and Roth, 2014) dictates that the least

possible amount of information should be published to complete a task. Classification predicts the class, a categorical variable, to which a new example belongs to. Thus, for this task, all that is really necessary to publish is the class label and not the model parameters. A private mechanism which conforms with the principle of least information and only releases private predictions is likely to handle privacy constraints more efficiently.

Such a private classification mechanism would work by first fitting a non-private model, using established training procedures; then injecting privacy noise only at the prediction stage. The private prediction procedure would utilize the exponential mechanism to release the class label, a discrete choice.

For Naive Bayes, the quality function of the exponential mechanism would be the log posterior probability of each class, computed using the true model parameters. To satisfy differential privacy, the log posterior could be either thresholded (Williams and McSherry, 2010) or its sensitivity could be derived. In tune with our intuition, the exponential mechanism would pick a noisy maximum of the posterior probability of a class.

For logistic regression, the probability of a class given a test example is determined by the logistic function (or the softmax) of the dot product of the weights and the test features. As such, prediction could be directly translated into the exponential mechanism: the quality function is the dot product of the weights and the features. As always, sensitivity analysis or thresholding would be in order.

Another possible direction for future work is Bayesian inference of the true model parameters. In particular, given noisy model parameters, learned by a differentially private fitting procedure, we could infer the posterior probability of the true parameters of the model. Then, prediction would be done by averaging the predictions of each possible true parameter weighted by this parameter's posterior probability given its noisy counterpart. Doing prediction in this way is statistically optimal. Although probabilistic inference with differential privacy has previously been detailed by (Williams and McSherry, 2010), we believe its application to private Naive Bayes and private logistic regression would be a useful contribution. It would stimulate the adoption of private machine learning methods in more practically-oriented communities.

As far as private feature selection is concerned, this work has proposed a solution for two-class problems. A natural continuation of this work is to provide heuristics for multi-class problems. Future work could also investigate more sophisticated feature selection procedures under the differential privacy constraint as well as provide further

empirical evidence for their performance benefit.

Chapter 6

Conclusion

Many of the proposed differentially private machine learning methods have nice asymptotic properties, such as asymptotically non-private performance guarantees, polynomial sample complexities, and computational efficiency. Unfortunately, a number of attempts to use these algorithms with realistic privacy settings and data have failed to achieve satisfactory results. The difficulty in applying private algorithms comes from their intrinsic dependence on dataset dimensionality. With this work we have demonstrated that private machine learning algorithms can be practical in realistic settings. Through feature selection we have shown that private classifiers can be easy to train, accurate, and with manageable data requirements. This work has further established that not all private machine learning algorithms are created equal. What are only training practicalities in non-private settings, such as a classifier's number of hyperparameters, reliance on regularization, and so on, are crucial for performance under privacy constraints, and should be determining factors in the choice of mechanism for real-world private data analysis.

Appendix A

Appendix 1: Datasets

In the following are listed the details of the several real datasets used throughout this work.

A.1 XwindowsDoc dataset

This dataset was originally used in (Murphy, 2012) to illustrate feature selection techniques with a naive Bayes classifier. The dataset is available from the pmtk3 download page (Doe, 2009).

It contains documents from two classes: X windows and MS Windows. The documents are represented by vectors of length 600, where the binary occurrences or words are recorded. The dataset contains 900 training samples, 900 test samples, and 600 dimensions. The classes are balanced both in the training and the test sets, so the majority baseline is 50%. A non-private Naive Bayes classifier obtains accuracy of 92% on the training set and 81% on the test set. Table A.2 shows a list of the most

Name	Train set size	Test set size	Number of features	Majority baseline
XWindowsDoc	900	900	600	50%
Mushrooms	5687	2437	112	52%
Adult	32561	16281	123	76%
Newsgroup	5687	2437	100	57%
Newsgroup2	4663	1998	100	81%
Newsgroup3	4663	1998	100	82%

Table A.1: Details of the real UCI datasets used in this work.

highest MI	MI
windows	0.215
microsoft	0.095
dos	0.092
motif	0.078
window	0.067
sun	0.045
code	0.044
win	0.044
xterm	0.040
server	0.040

Table A.2: Reducing the dimensionality to use only the top 10 most informative features.

informative features sorted by their mutual information with the class label. Table A.3 shows the number of times the 10 most informative features are on in the two classes.

A.2 Mushrooms dataset

The UCI Mushrooms dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended, the latter two classes combined.

Each nominal attribute is expanded into several binary attributes, as per one-hot encoding. The original attribute number 12 has missing values and is not used. The number of training samples is 5687 and number of test samples is 2437. The number of resulting features is 112.

A.3 Newsgroup1 dataset

This is a binary classification dataset constructed by using the first two classes of the 20newsgroups data from (Lang, 2014). It contains binary occurrence data for 100 words (features) across 16242 postings.

Variable	$N^{(1)}$	$N^{(0)}$
windows	56	288
microsoft	6	107
dos	10	115
motif	80	2
window	137	32
sun	65	7
code	82	15
win	5	95
xterm	40	0
server	74	13

Table A.3: The number of occurrences of the top 10 most informative features in both classes. This shows us how sparse the feature are.

A.4 Newsgroup2 dataset

This dataset only uses the third and fourth classes of 20newsgroups to form a binary classification problem. This dataset is highly imbalanced, with 856 training examples of class 1 and 3807 examples of class 2. The imbalance was created by taking only a portion of the documents in class 3 of the original 20newsgroup dataset.

A.5 Newsgroup3 dataset

This dataset only uses the third and fourth classes of 20newsgroups to form a binary classification problem. This dataset is highly imbalanced, with 849 training examples of class 1 and 3814 examples of class 2. The imbalance was created by taking only a portion of the documents in class 2 of the original 20newsgroup dataset.

A.6 Adult Dataset

The Adult dataset from the UCI ML Repository is moderately-sized dataset that contains demographic information about approximately 47 000 individuals. The task is to predict whether the annual salary of individuals is below $50 000, based on 105 features, such as occupation, education, age, etc. This set is imbalanced: the fraction of positive labels

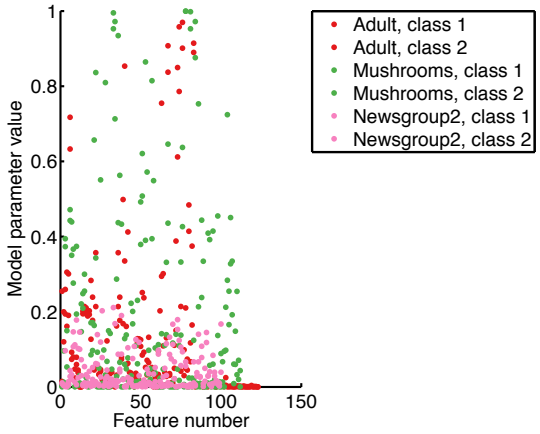


Figure A.1: Relative sparsity of the Adult, Mushrooms, and Newsgroup2 datasets. We judge the sparsity by the model parameters obtained by fitting a non-private Naive Bayes classifier. On the horizontal axis is the feature number. The corresponding model parameter is on the y axis. The Mushrooms dataset is the least sparse, and Newsgroup2 and Adult are fairly sparse.

is around 1/4, i.e. the majority baseline is 76%.

All samples with missing values were removed and each categorical attribute was represented with 1-of-K encoding. Each column was normalized such that the maximum is 1 and each row, such that it's norm is *at most* 1.

A.7 Toy datasets

To perform controlled experiments with datasets of different sizes and dimensionality, it was important to focus on a really simple problem to start with.

We create a simple two-class dataset, where data for each class was sampled from a Gaussian distribution. The data is non-linearly separable, so that the effects of the perturbation of the decision boundary are more obvious. In particular,

$$x^{(n)} \in C_1 \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$$
$$x^{(n)} \in C_2 \sim \mathcal{N}(\alpha\mathbf{1}, \mathbb{I})$$

where α is a constant that is used to control the amount of overlap between the two classes. Alpha was picked so that the Bayes classifier achieves 90% accuracy:

$$\alpha = \sqrt{\frac{4}{D}} F^{-1}(0.9) \tag{A.1}$$

where $F^{-1}(0.9)$ is the 0.9-the quantile of the normal distribution.

The toy dataset thusly created is balanced, meaning the prior probability of each class is around 50%. The samples generated with this procedure are then split into three sets of equal sizes and used for training, validation, and testing, respectively.

A.8 Preprocessing

To use with Bernoulli Naive Bayes, whenever necessary, features were converted to binary by discretizing and using 1-of-K encoding.

For logistic regression, categorical variables were 1-of-K encoded. Additionally, each feature was scaled to fall within the $[-1,1]$ range by:

$$\mathbf{x'_d} = \frac{\mathbf{x_d} - \min(\mathbf{x_d})}{\max(\mathbf{x_d}) - \min(\mathbf{x_d})} \tag{A.2}$$

and then translating to $[-1,1]$. Finally, to comply with the assumptions of the output and objective perturbation methods, detailed in Chapter 4, all examples were scaled so that their Euclidean norm is at most 1.

Appendix B

Appendix 2: Private Logistic Regression Scalability

This appendix demonstrates the poor scalability with dimensionality of two algorithms for private logistic regression: the objective perturbation algorithm and the functional mechanism, both discussed at length in Chapter 4. We further improve the functional mechanism here to correct the sensitivity of the approximation to the objective function reported in (Zhang et al., 2012). Experiments are performed on the toy datasets as detailed in Section A.7, which allow variable settings for size and dimensionality.

B.1 Functional Mechanism

Proposed by Zhang et al. (2012), the functional mechanism for private logistic regression, referred to from now on as simply the functional mechanism, does not impose the strong convexity requirements of the Private ERM algorithm described earlier. The method works with non-regularized logistic regression.

Briefly, the method works by creating a Taylor approximation to the optimization function and adding noise to its coefficients. The objective function is then perturbed by applying the Laplace mechanism, where noise is drawn from a Laplace distribution with scale the sensitivity of the objective function over the privacy parameter, i.e Δ/ε.

The private logistic regression objective function resulting from their method is as follows:

$$J_{approx}(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{2} \frac{f_1^{(k)}(0)}{k!} (x_i\theta)^k - \left(\sum_{n=1}^{N} y_i x_i^T \right) \theta, \tag{B.1}$$

where the coefficients result from the Taylor expansion.

B.1.0.0.1 Improving the reported sensitivity The sensitivity of the approximate objective function is:

$$\Delta = 2\max_{x}\left(\frac{f_1^{(1)}(0)}{1!}\sum_{d=1}^{D}x_d + \frac{f_1^{(2)}(0)}{2!}\sum_{j},lx_jx_l + y\sum_{d}x_d\right) \leq 2\left(\frac{D}{2}+\frac{D^2}{8}+D\right) = \frac{D^2}{4}+3D \tag{B.2}$$

We refer to the Functional Method using this sensitivity as FMsens1.

The bound on the sensitivity that was reported can be tightened by taking into account the assumption this algorithm makes that feature vectors are normalized to unit length. In more detail, all samples in the dataset are normalized so that the Euclidean norm of a sample is at most one:

$$\sqrt{\sum_{d=1}^{D}x_d^2} \leq 1 \tag{B.3}$$

In addition, the square root function is concave. Jensen's inequality applies and states that a concave transformation of a mean is greater than or equal to the mean after the concave transformation:

$$\varphi\left(\mathbb{E}[X]\right) \geq \mathbb{E}\left[\varphi(X)\right]. \tag{B.4}$$

From Jensen's inequality and assuming that x_d is non-negative:

$$\sqrt{\frac{\sum_{d=1}^{D}x_d^2}{D}} \geq \frac{\sum_{d=1}^{D}x_d}{D} \tag{B.5}$$

Simplifying and applying B.3 we obtain:

$$\sum_{d=1}^{D}x_d \leq \sqrt{D}\sqrt{\sum_{d=1}^{D}x_d^2} \leq \sqrt{D} \tag{B.6}$$

Contrast this to the D bound that was present in Equation B.2. Similarly, we can bound the other terms in that equation to obtain an improved bound on the sensitivity:

$$\Delta = \frac{D}{4}+3\sqrt{D}. \tag{B.7}$$

The Functional Mechanism with this bound on the sensitivity is referred to as FMsens2 from now on.

We now perform experiments with the Private ERM mechanism, as well as the Functional Mechanism with both the original and improved bounds on the sensitivity of the objective function.

B.2 Experimental Setup

Convergence criterion We consider an algorithm converged when it achieves average accuracy that is within a certain range from the MAP of the logistic regression. The tolerance used in reporting the results is around 2% meaning we consider the private algorithm converged when its accuracy is within 2% of the MAP of logistic regression. This is not the best criterion of convergence for a number of reasons. To start with in practice 2% loss in accuracy may not be permissible. Even more importantly, due to the variance in the noise, it is possible that we accidentally arrived within the desired distance from the MAP. Thus, it is necessary to correct for any inaccuracies that may have arisen. This is done by rounding up to the next power of 2 during the first pass that finds a crude estimate (explained below) and rounding up to the upper bound of the interval in the finer estimation of N (also explained below). Despite these shortcomings, this measure is easy to work with and for the purpose of finding out a crude relationship between the number of examples and the accuracy achieved seems like a good compromise.

B.3 Results

Here we experiment with several different dimensionality settings and report the results. Tables B.2 - B.5 show the number of data samples needed for the different algorithms to converge to the ML. Table B.1 shows the best accuracies we could hope for and the maximum amount of data that is available to the algorithms in all the settings. Results were obtained by averaging the results over 10 runs of the randomized algorithms. Regularization parameters are chosen by cross-validation on a specially allocated cross-validation set. The privacy parameter is taken to be 0.1, which is a very common setting.

We now look at how the different algorithms performed under the various dimensionality settings.

Table B.2 shows the number of samples needed for LR to reach the MLE. We see that in small dimensionality settings, relatively small regularization is needed, which is not surprising - our model cannot overfit these datasets too much, unless they are linearly separable, which is not the case due to our setup.

Private ERM Table B.3 shows the number of samples needed for the Private ERM logistic regression algorithm to reach roughly the same accuracy as the non-private

logistic regression, together with the regularization parameters picked. We see that these are quite high compared to the ones for logistic regression. The algorithm has preferred the first branch of the conditional, which is the one that uses more relaxed privacy budget (apart from $D = 1$) and larger regularization. The privacy constraints are "shared" by the increase in regularization and the noise added.

The bigger regularization parameter says that we are more sure the weights θ are around the new mean of the prior ($-1/\lambda b$), and we don't rely on the data as much as before. The increase in regularization means that we will reach the MLE more slowly (i.e. by using more data).

The final column shows how many times more data we need to provide to the private logistic regression in order to get close to the MLE value of logistic regression. We see that Private ERM is fairly well-behaved: the multipliers grow linearly with the dimensionality of the data. Since the results are not exact, we can't give a precise constant of growth, but it's not unreasonable to guess 2. The main point is that there is no exponential increase in the number of samples needed when we increase the dimensionality. Having said that, significant amount of additional data is necessary for good performance, making this algorithm impractical in a lot of real-world settings.

Functional Mechanism Table B.4 shows the results for FM with the original reported sensitivity. Even for trivial dimensionality settings this method requires unreasonable amounts of additional data: for $D = 5$ we need 256 times more data than normal to reach values close to the MLE. For bigger dimensions the standard error increases and we no longer have a very reliable estimate of the number of samples needed, but it is easy to see that this would be very high. These results are not surprising given the quadratic reliance of the sensitivity on the dimensionality of the data. This algorithm is impractical for any dimensionality bigger than 10.

Table B.5 shows the results for FM with the improved sensitivity. We get improvements for smaller dimensions and even reasonably good performance for up to $D = 20$. This is more encouraging than the results from FMsens1, but still does not beat the Private ERM method. Where it might make a difference is in the running time. Because of the quadratic approximation, the minimum can be found analytically, which is faster than using an iterative approach.

We run two more experiments with PrivateERM and FMsens2 to confirm these expectations. We give the algorithms even more training data. Table B.6 shows the results for PrivateERM with the smaller value of ε. Indeed the multipliers are much larger - 16 times bigger than the ones for $\varepsilon = 0.1$. Incidentally, 16 is the nearest power

	Truth		
D	λ	All N	Accuracy
1	1e-06	1024	88.0
2	1e-06	4096	91.9
5	1	4096	89.9
10	1e+02	4096	89.1
20	1e+03	8192	90.0
50	1	1048576	89.9
100	1e+03	524288	90.0
1000	1e+03	1048576	90.0
1500	1e+03	1048576	90.0

Table B.1: True accuracy and total number of samples. This table shows the maximum data that was available to all algorithms in training. The accuracy is the best that we could hope for on every dataset.

	LR		
D	λ	N	Accuracy
1	1e-06	16	88.0
2	0.1	16	91.9
5	1	16	88.3
10	1	32	87.6
20	10	128	88.6
50	1	256	88.0
100	10	512	88.3
1000	1e+03	16384	88.1
1500	1e+03	32768	88.0

Table B.2: MAP results for Logistic Regression on various datasets. We show the minimum number of samples that get us close to the MLE, accurate to a power of 2.

PrivateERM					
D	λ	N	Accuracy	Standard Error	Multipliers
1	0.0001	256	88.0	0	16
2	1e+02	512	90.8	0.7	32
5	1e+02	1024	88.3	0.4	64
10	1e+03	2048	87.3	0.4	64
20	1e+03	8192	87.9	0.3	64
50	10	131072	88.0	0.2	512
100	1e+03	65536	88.1	0.1	128
1000	1e+03	1048576	79.1	0.2	64
1500	1e+03	1048576	70.5	0.3	32

Table B.3: Number of samples needed for PrivateERM to get close to the MLE of LR for $\varepsilon = 0.1$, accurate to a power of 2. The multipliers show many times more data is required compared to Logistic Regression.

FMsens1				
D	N	Accuracy	Standard Error	Multipliers
1	256	88.0	0	16
2	1024	90.7	0.9	64
5	4096	85.9	0.9	256
10	4096	61.6	2.7	128
20	8192	55.8	3.6	64
50	1048576	66.9	2.3	4096
100	524288	50.7	0.9	1024
1000	1048576	49.6	0.4	64
1500	1048576	50.1	0.3	32

Table B.4: Number of samples needed for FMsens1 to get close to the MLE of LR. Regularization is ignored by the algorithm, since it is taken to be 4 times the standard deviation of the Laplace noise added. We see that the multipliers here are much worse when the dimensionality grows.

FMsens2				
D	*N*	Accuracy	Standard Error	Multipliers
1	128	88.0	0	8
2	512	89.9	1.2	32
5	4096	86.9	0.7	256
10	4096	82.4	1.3	128
20	8192	68.4	1.8	64
50	1048576	75.2	3.1	4096
100	524288	68.1	1.6	1024
1000	1048576	50.9	0.4	64
1500	1048576	50.6	0.2	32

Table B.5: Number of samples needed for FMsens2 to get close to the MLE for $\varepsilon = 0.1$. Regularization is again ignored. Multipliers are worse than those for the Private ERM, but we get better performance where we can.

of 2 bigger than 10 and since our results are exact to powers of 2, this confirms our expectations. The situation is similar for FMsens2 as shown in table B.7.

B.4 Discussion

In very low dimensions the recommended method for private logistic regression is the Functional Mechanism with improved sensitivity: it needs roughly the same amount of additional data as the Private ERM and the optimum can be found analytically. In higher dimensions, we suggest using the Private ERM method.

In these experiments regularizers were selected by non-private cross-validation. To preserve end-to-end privacy some of the privacy budget should be spent on cross-validation (Chaudhuri and Vinterbo, 2013b).

To summarize, these are not great results for differential privacy. Although in the limit of infinite data all of these methods approach the MLE (ignoring the approximation error in the Function Mechanism), the amount of additional data required is prohibitive. Even for these relatively easy problems (balanced classes, low-dimensional dense features) we need too much additional data to get reasonable performance.

PrivateERM					
D	λ	N	Accuracy	Standard Error	Multipliers
1	0.01	4096	89.9	0.0	256
2	1e-05	8192	88.2	0.7	512
5	1e+03	16384	88.3	0.3	1024
10	1e+03	32768	88.3	0.3	1024
20	1e+03	131072	88.6	0.2	1024

Table B.6: Number of samples needed for PrivateERM to get close to the MLE, for $\varepsilon = 0.01$. Multipliers for this epsilon are 16 times larger than those for the smaller $\varepsilon = 0.1$.

FMsens2				
D	N	Accuracy	Standard Error	Multipliers
1	4096	89.9	0.0	256
2	8192	88.1	0.8	512
5	65536	88.3	0.3	4096
10	131072	88.2	0.5	4096
20	262144	77.5	1.6	2048

Table B.7: Number of samples needed for FMsens2 to get close to the MLE. We also see an increase of around 16 times in these multipliers compared to multipliers for FMsens2 with $\varepsilon = 0.1$.

PrivateERM					
D	λ	N	Accuracy	Standard Error	Multipliers
1	0.1	512	88.1	1.2	32
2	0.01	2048	91.0	0.3	128
5	1e+02	2048	88.0	0.4	128
10	1e+02	4096	84.2	1.2	128
20	1e+02	8192	82.7	0.9	64

Table B.8: Number of samples needed for PrivateERM to get close to the MLE for the second type of feature scaling. These results are even worse for differential privacy: the algorithm gives up faster, and requires more data even in lower dimensions.

Bibliography

Prateek Jain 0002 and Abhradeep Thakurta. Differentially private learning with kernels. In *ICML (3)*, volume 28 of *JMLR*, pages 118–126, 2013. URL http://dblp.uni-trier.de/db/conf/icml/icml2013.html#0002T13.

Prateek Jain 0002, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *COLT*, volume 23 of *JMLR*, pages 24.1–24.34, 2012. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlrp23.html#JainKT12.

Atockar. Riding with the stars: Passenger privacy in the NYC taxicab dataset, 2014. URL http://research.neustar.biz/author/atockar/. [Online; accessed 12-Dec-2014].

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008. doi: 10.1145/1374376.1374464.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. In *JACMS*, volume 60 of *JACM*, page 12, 2013.

Justin Brickell and Vitaly Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *KDD*, pages 70–78, 2008. doi: 10.1145/1401890.1401904. URL http://doi.acm.org/10.1145/1401890.1401904.

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008. URL http://dblp.uni-trier.de/db/conf/nips/nips2008.html#ChaudhuriM08.

Kamalika Chaudhuri and Staal A. Vinterbo. A stability-based validation procedure for differentially private machine learning. In *NIPS*, pages 2652–2660, 2013a. URL http://dblp.uni-trier.de/db/conf/nips/nips2013.html#ChaudhuriV13.

Kamalika Chaudhuri and Staal A. Vinterbo. A stability-based validation procedure for differentially private machine learning. In *NIPS*, pages 2652–2660, 2013b.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. In *JMLR*, volume 12, pages 1069–1109, July 2011. URL http://dl.acm.org/citation.cfm?id=1953048.2021036.

Kamalika Chaudhuri, Anand D. Sarwate, and Kaushik Sinha. Near-optimal algorithms for differentially-private principal components. In *CoRR*, volume abs/1207.2812, 2012. URL http://dblp.uni-trier.de/db/journals/corr/corr1207.html#abs-1207-2812.

Jacek Czerniak and Hubert Zarzycki. Application of rough sets in the presumptive diagnosis of urinary system diseases. In *Artificial Intelligence and Security in Computing Systems*, volume 752 of *The Springer International Series in Engineering and Computer Science*, pages 41–51, 2003. doi: 10.1007/978-1-4419-9226-0_5. URL http://dx.doi.org/10.1007/978-1-4419-9226-0_5.

Ringo Doe. PMTK, June 2009. URL https://github.com/probml/pmtk3/.

Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag. URL http://dl.acm.org/citation.cfm?id=1791834.1791836.

Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *STOC*, pages 371–380, 2009.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(34):211–407, 2014. doi: 10.1561/0400000042. URL http://dx.doi.org/10.1561/0400000042.

Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis. In *STOC*, pages 11–20, 2014. doi: 10.1145/2591796.2591883. URL http://doi.acm.org/10.1145/2591796.2591883.

Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. In *CoRR*, volume abs/1310.3197, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1310.html#ErlichN13.

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security*, pages 17–32, 2014.

Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010. doi: 10.1145/1835804.1835868. URL http://doi.acm.org/10.1145/1835804.1835868.

Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1170–1178, 2014. URL http://jmlr.org/proceedings/papers/v32/gaboardi14.html.

Quan Geng and Pramod Viswanath. The optimal mechanism in differential privacy: Multidimensional setting. In *CoRR*, volume abs/1312.0655, 2013. URL http://arxiv.org/abs/1312.0655.

Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.*, 41(6):1673–1693, 2012.

Abhradeep Guha Thakurta and Adam Smith. (Nearly) optimal algorithms for private online learning in full-information and bandit settings. In *NIPS*, pages 2733–2741. 2013. URL http://goo.gl/f9bP2J.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. In *JMLR*, volume 3, pages 1157–1182, March 2003.

Rob Hall. New statistical applications for differential privacy, 2013.

M. Hardt and G.N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 61–70, Oct 2010. doi: 10.1109/FOCS.2010.85.

Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. A practical differentially private random decision tree classifier. In *TDP*, volume 5, pages 273–295, April 2012. URL http://dl.acm.org/citation.cfm?id=2207141.2207145.

Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *CoRR*, volume abs/1109.0105, 2011. URL http://arxiv.org/abs/1109.0105.

Zhanglong Ji, Zachary Chase Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. In *CoRR*, volume abs/1412.7584, 2014. URL http://arxiv.org/abs/1412.7584.

Daniel Kifer, Adam D. Smith, and Abhradeep Thakurta. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. In *COLT*, volume 23 of *JMLR Proceedings*, pages 25.1–25.40, 2012. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlrp23.html#KiferST12.

Aleksandra Korolova. Privacy violations using microtargeted ads: A case study. In *IEEE Workshop on Privacy Aspects of Data Mining*, pages 474–482, 2010.

Ken Lang. 20 newsgroups, 2014. URL http://qwone.com/~jason/20Newsgroups/.

Jaewoo Lee, Yue Wang, and Daniel Kifer. Maximum likelihood postprocessing for differential privacy under consistency constraints. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 635–644, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783366. URL http://doi.acm.org/10.1145/2783258.2783366.

Jing Lei. Differentially private m-estimators. In *NIPS*, pages 361–369, 2011. URL http://dblp.uni-trier.de/db/conf/nips/nips2011.html#Lei11.

Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115. IEEE, 2007. URL http://dblp.uni-trier.de/db/conf/icde/icde2007.html#LiLV07.

Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. In *TKDD*, volume 1, March 2007. doi: 10.1145/1217299.1217302. URL http://doi.acm.org/10.1145/1217299.1217302.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007. doi: 10.1109/FOCS.2007.41. URL http://dx.doi.org/10.1109/FOCS.2007.41.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, 2008. doi: 10.1109/SP.2008.33. URL http://dx.doi.org/10.1109/SP.2008.33.

Natalie Newman. Netflix sued for largest voluntary privacy breach to date, 2009. URL http://goo.gl/pAoJII.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848. MIT Press, 2002. URL http://goo.gl/fgQYoA.

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007. doi: 10.1145/1250790.1250803. URL http://doi.acm.org/10.1145/1250790.1250803.

Sewoong Oh and Pramod Viswanath. The composition theorem for differential privacy. In *CoRR*, volume abs/1311.0776, 2013. URL http://arxiv.org/abs/1311.0776.

Carl Edward Rasmussen. Minimize a differentiable multivariate function., December 2014. URL http://learning.eng.cam.ac.uk/carl/code/minimize/minimize.m.

Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. In *CoRR*, volume abs/0911.5708, 2009. doi: http://arxiv.org/abs/0911.5708.

Ben Stoddard, Yan Chen, and Ashwin Machanavajjhala. Differentially private algorithms for empirical machine learning. In *CoRR*, volume abs/1411.5428, 2014. URL http://arxiv.org/abs/1411.5428.

Latanya Sweeney. K-anonymity: A model for protecting privacy. In *IJUFKS*, volume 10, pages 557–570, October 2002. doi: 10.1142/S0218488502001648. URL http://dx.doi.org/10.1142/S0218488502001648.

J. Vaidya, B. Shafiq, A. Basu, and Yuan Hong. Differentially private naive bayes classification. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 571–576, Nov 2013. doi: 10.1109/WI-IAT.2013.80.

Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *ACM CCS*, CCS '09, pages 534–544. ACM, 2009. doi: 10.1145/1653662.1653726. URL http://doi.acm.org/10.1145/1653662.1653726.

Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. In *JASA*, volume 105, pages 375–389, 2009. doi: 10.1198/jasa.2009.tm08651.

Wikipedia. Aol search data leak, Wikipedia, the free encyclopedia, 2004. URL http://en.wikipedia.org/wiki/AOL_search_data_leak. [Online; accessed 12-Dec-2014].

Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *NIPS*, pages 2451–2459, 2010.

Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: Regression analysis under differential privacy. *PVLDB*, 5(11):1364–1375, 2012.

Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. In *ACM SIGMOD/PODS*, SIGMOD '14, pages 1423–1434. ACM, 2014. doi: 10.1145/2588555.2588573. URL http://doi.acm.org/10.1145/2588555.2588573.